NEC

PC-6001

みんなで使おうBASIC



PC-6001
みんなで使おうBASIC



NEC

ご注意

(1) 本書の内容の一部または全部を無断で転載することは禁止されています。

(2) 本書の内容に関しては，将来予告なしに変更する事があります。

(3) 本書は内容について万全を期しておりますが，万一ご不明な点や，あやまり，記載もれなど，お気付きのことがありましたら，ご連絡下さい。

(4) 運用した結果の影響については，(3)項にかかわらず，責任を負いかねますので，ご了承下さい。

まえがき

この本は，これから初めてパーソナルコンピュータPC-6001を使おうとしているきみのために書かれたものです．

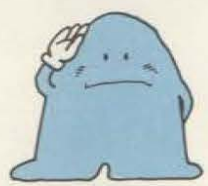
PC-6001は，コンピュータとしての便利な計算能力の他に，きれいな絵を描いたり，美しいメロディーを奏でるなど，たくさんの楽しい機能を持っています．この本を読んでいくうちに，きみは自分でこれらの機能を自由に使いこなすことができるようになるでしょう．

まず第１部の初級編では，PC-6001本体の取り扱い方と，PC-6001を操るためのBASIC言語の基礎について説明します．そして第２部の上級編では，さらに便利なBASICの応用を紹介していきましょう．

この本を読むときには，PC-6001を目の前に置いて，実際にきみ自身の手で結果を確かめながら進んでください．そうすれば難しいことはひとつもないはずです．もしも知らない言葉や忘れてしまったことが出てきたら，そのときはすぐに目次や索引のページを開いて，意味を調べてください．そのたびにきみの力は少しずつ増していくのですから．

また，きみがこの本を卒業して自分の力でPC-6001を動かせるようになったときでも，何か困ったことやわからないことがあったら，この本のことを思いだしてください．いろいろなヒントをあちこちのページで見つけることができるでしょう．

PC-6001はきみを待っています．さあ，扉を開けてください．



もくじ

第1部　初級編



第1章　スタンバイ　11

第2章　キーボードに慣れよう　15

第2部 上級編



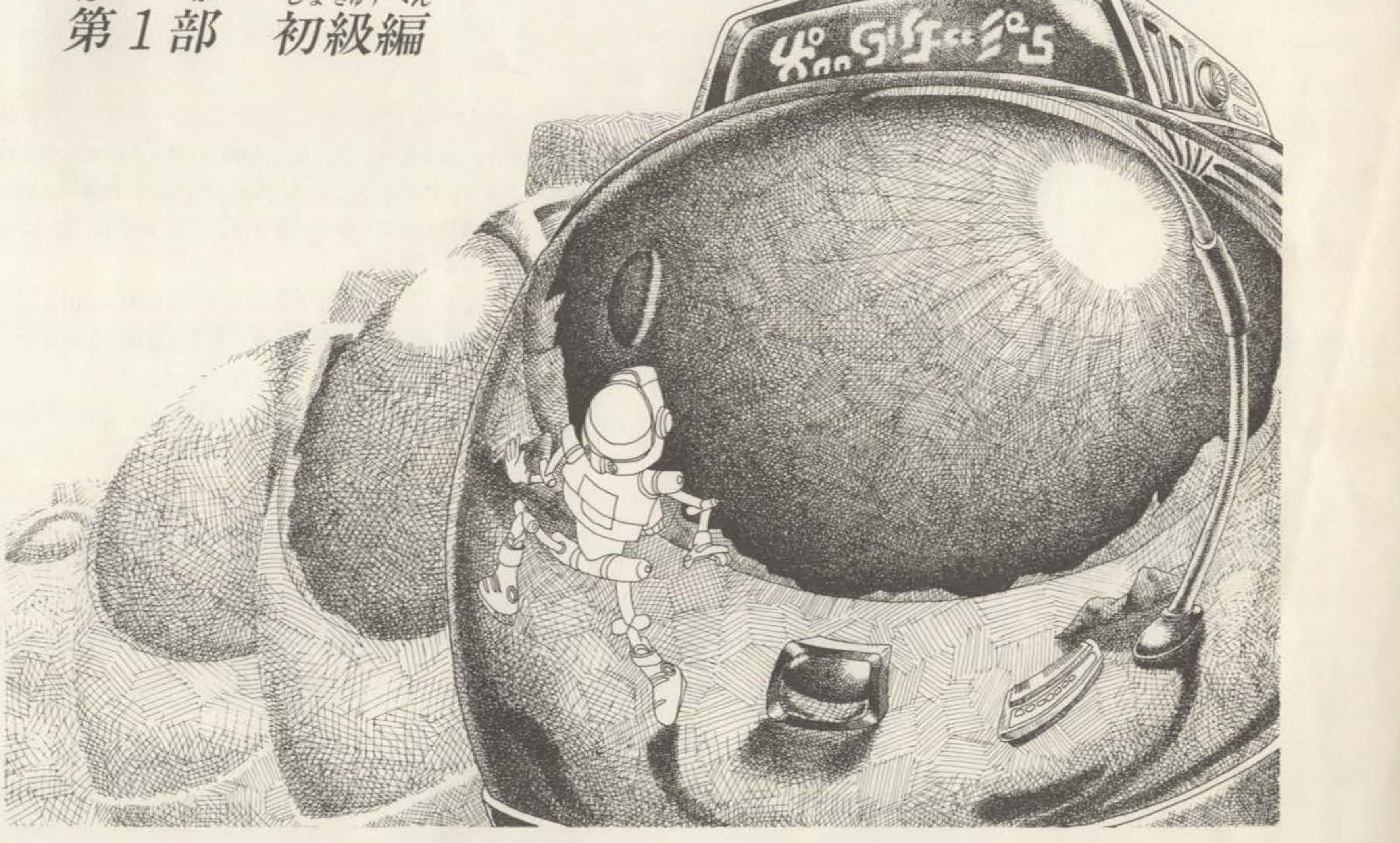
第2章　文字だって思いのまま　225

第3章　さらに便利な命令　247

第4章　ジョイステックと，プリンタと　259

第5章　コンピュータのことば　267

第1部　初級編





第1部初級編の始まりです．

最初は，PC-6001をどう扱ったらいいのか，どこを押して，どこに触れれば正しく動いてくれるのか，ということについて話を進めていきます．このとき，恐がらないで自分から積極的にPC-6001にさわってください．使えば使うほどPC-6001がよくわかってきます．ここで充分に使いこなしておけば，以後この本を読むのがとても楽になるはずです．

そして次に，BASIC 言語について説明します．この言語を覚えれば，きみはPC-6001に思いどおりの仕事をさせることができるのです．第1部ではまだ基礎的な使い方しか説明しませんが，これだけでもPC-6001の持つすべての機能を楽しむことができるでしょう．

パラパラとページをめくると，いろいろ難しそうな文字が目に入るかもしれませんが，心配はしないでください．この本を最初から順々に読んでいけば，BASIC にはひとつも難しいところはありません．ただし，目で読むだけではいけません．とにかく自分の指で試してみること，これが大事です．

なお，この本の中に出てくる BASIC プログラムのリストは，すべて横1行につき32文字で，PC-6001がテレビの画面に表示する形とまったく同じになっています．プログラムを打ちこむ場合には，これをおおいに参考にして，正確な作業を行ってください．

それではいよいよ第1部初級編に入りましょう．

第1章　スタンバイ



コンピュータといっても，電気の力で働く機械にはかわりません．まずは“電源の入れ方”から，説明しましょう．すべてのお話は，そこから始まります．

1－1　用意するもの



① PC-6001　(以後，単に“PC”とも呼びます．)
②家庭用カラーテレビ　(以後，“ディスプレイ”とか“CRT”ともいいます．)
③ PC-6001とテレビをつなぐコード

②のカラーテレビは，白黒テレビでも代用できますが，そのときには，せっかくの“カラーグラフィックス”を楽しむことができなくなります．ですから，なるべくならカラーテレビやコンピュータ専用のカラーディスプレイを手に入れたほうが良いでしょう．

1－2　PCのセッティング



さて，準備したPCとテレビを，机やテーブルのような，平な場所に置きましょう．まず，テレビを少し奥のほうに置いて，こちらに画面を向けます．その手前に，PCを置いてください．

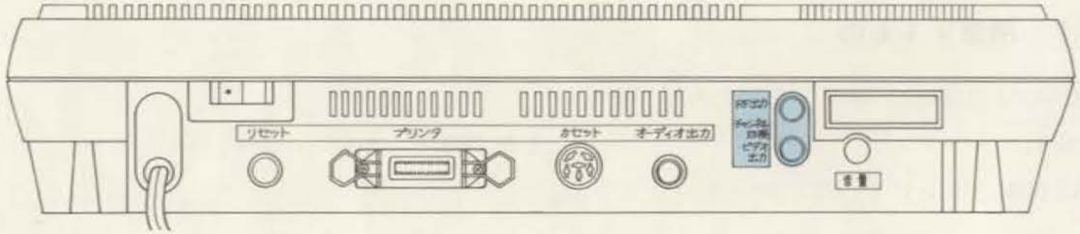
なお，コンピュータは水分には弱いので，水がかかるような場所はいけません．また，PCの上にコーヒーやジュースなどをこぼさないようにしてください．

1－3　PC-6001とテレビの接続

付属の接続コードは，専用ディスプレイのもので，両端がピンプラグになっています．家庭用の

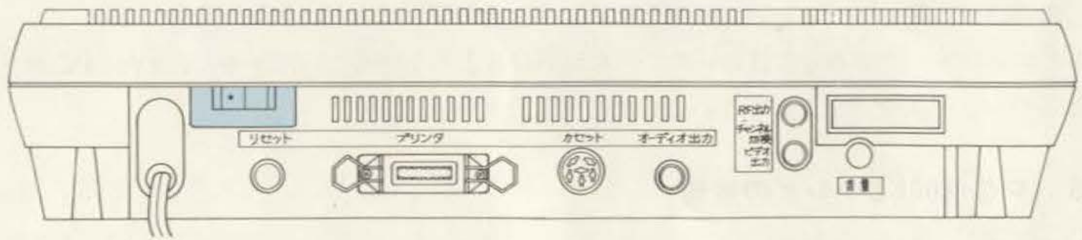
テレビに接続するためには，一方の端を切って裸にむくか，家庭用テレビ接続ケーブル PC-6091を使ってください．そして，むいた端をテレビの VHF 用アンテナ端子につないでください．これは，テレビによっては，300Ω 平行フィーダ用と75Ω 同軸ケーブル用の 2 つがありますが，75Ω 同軸ケーブル用端子につなぎます．詳しくは，PC-6001付属のマニュアルをみてください．

接続コードの，ピンプラグが付いている端は，PC-6001につなぎます．PC の背中を見てください．右のほうに，縦に並んだ 2 つのピンジャックがあります．この上のほうのピンジャックに，ピンプラグをさしこんでください．

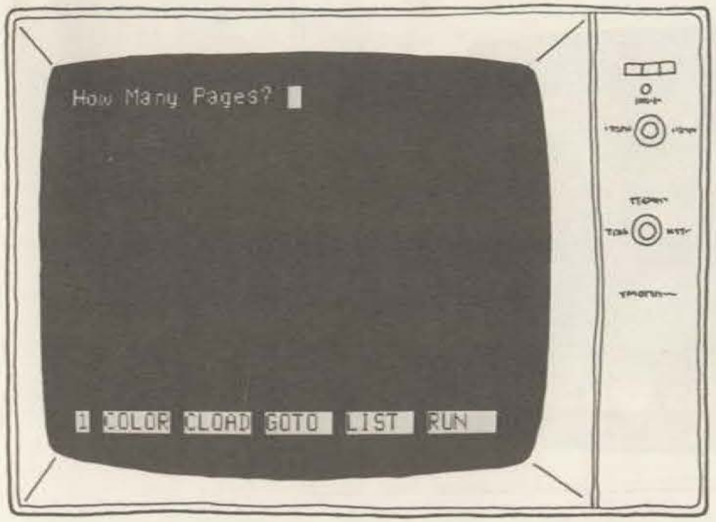


1－4　パワースイッチをON(オン)する

初めに，テレビのスイッチを入れ，第 1 チャンネル，または第 2 チャンネルに合わせます．次に PC の背面にあるパワースイッチを ON(オン) にしてください．



このとき，テレビの画面は写真のようになっているはずです．



もし，これと同じ画面にならなかった場合は，テレビの微同調ツマミを回してみてください．それでもだめなときは，チャンネルの1と2を切り替えて調整してください．さらにだめなときは，もう1度PCとテレビの接続を確かめてください．どちらかのチャンネルで必ずこれと同じ画面がでます．詳しくは，PC-6001に付属している取扱説明書をみてください．

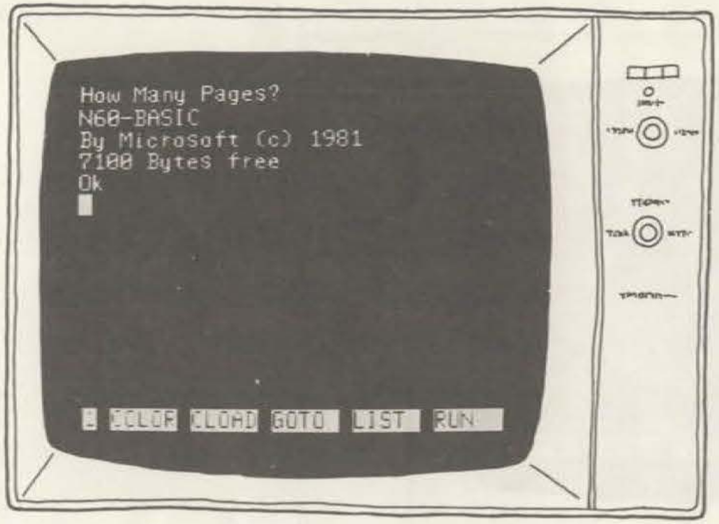
この画面がでましたら，テレビの輝度調整ツマミや，色調整ツマミを回して，画面上の文字が読みやすくなるよう調節しておきましょう．見づらい画面では，目がすぐ疲れてしまいますから．

1－5　最後のひと押し



PCの上面には，たくさんのボタンが並んでいます．その中で向かって右の端近くに，**RETURN** と書いてある，横に細長いものがあります．それを押してください．

そうすると，画面は次のページの写真のように変わります．



これで，すべての準備が終わりました．
いよいよ次の章から，PC-6001が動き出します．



第2章　キーボードに慣れよう



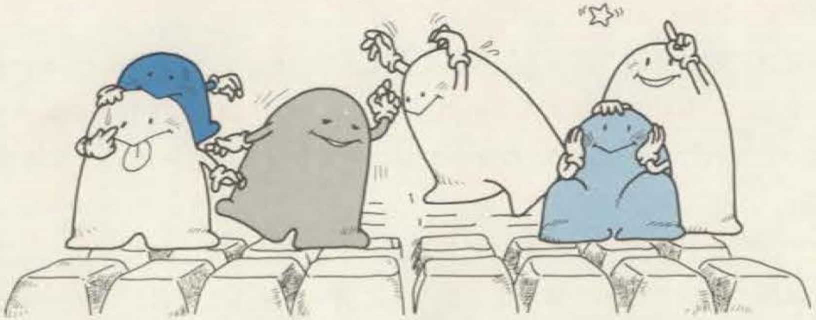
たとえば，きみが友達と話をしようとするときに，どうしてもこれだけは絶対に必要だ，ということはなんだと思いますか．まず，きみの考えていること，思っていることを，相手に伝えてやらなくてはなりません．また，相手の思っていることも，やはりきみに伝わってこなくてはなりません．もしこのどちらがなくなっても，言葉は一方通行になるばかりです．そんなふうでは，互いに相手を知ることはできないでしょう．

相手がコンピュータであっても，同じことです．きみの命令がうまくPCに伝わらなくても，PCのメッセージをきみが見逃しても，PCを上手に扱うことはできません．だからまず，キーボードとディスプレイに，よくなじむこと，それがPCを楽しむためのコツです．

2－1　とにかく触ってみよう

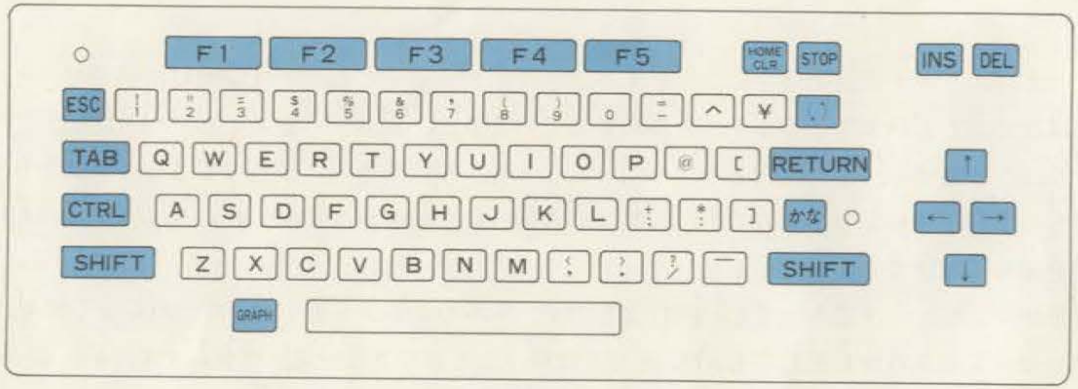
PC-6001の上には，いろいろな字や記号が書かれている押しボタンのようなものが，たくさん並んでいます．この装置全体を“キーボード”といい，ひとつひとつの押しボタンを“キー”といいます．







それぞれのキーには，みな別々の働きがありますが，それでも大きく2つの種類に分けられます．押すとテレビ画面に字が現れる“一般キー”と，それ以外の“特殊キー”です．下図で白くなっているのが一般キーで，実際のPCでは，灰色をしています．また下図で青色に塗られているのが特殊キーで，実物は橙 色をしています．





一般キーを押すと，どんなことが起こるのか，実際に試してみましょう．まず最初に，A のキーを押してみてください．

“ポッ！”というような音がして，画面にはアルファベットの小文字の“a”が現れました．他のキーも，いろいろ押して結果を見てください．各キーに対して，それぞれ決まった文字が画面に表示されるでしょう．このようにキーを押して，PCに何かを指示してやることを，“キーインする”とか，“キーを打つ”などといいます．



キーインするたびに聞こえる“ポッ！”という音は，“クリック音”といい，キーがちゃんと押されているかどうかを，いちいち確認するためのものです．もしも，あまり音が大きすぎるとか，小さくて聞こえないと思ったときは，PCの背中の面にあるツマミを回せば，音量を調整することもできます．

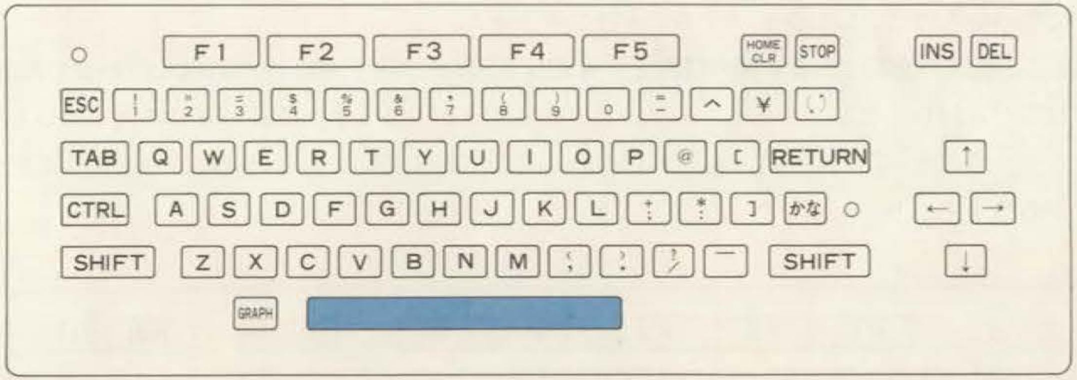
一般キーの中で，アルファベットの書いてあるものを押すと，そのアルファベットの小文字が出てきます．それ以外のキーでは，2個の文字がキーに書いてあるものは，下の文字が現れ，キーの

面に1個だけの字が書いてあるものは，その1個の文字が現れます．

[A] ……… “a”　（アルファベットのキーは，小文字が出る）
[!/1] ……… “1”　（2個の文字が書いてあるキーは，下の文字が出る）
[¥] ……… “¥”　（アルファベット以外で1個の文字が書いてあるキーは，その文字が出る）

キーボードのいちばん下にある細長いキーは，一般キーの中でもひとつだけ変わった形をしていますね．このキーは“スペースキー”，またの名を“スペースバー”といい，このキーを押すと，“空白”（1文字分のすき間）が現れます．空白は，他の文字よりも使う回数が多いので，押しやすいように，このとおり特別に大きくなっているのです．





2－2　カーソルを動かす

いろいろな文字を打ちこんでいくとき，いつでも文字の後ろで白い四角形が点滅していることに気がつきましたか．この四角形を“カーソル”といいます．カーソルは次に文字が表示される位置を示す印です．文字をひとつキーインすると，それまでカーソルがあった位置に，打ちこんだ文字が現れ，カーソル自身は1文字右へ移ります．





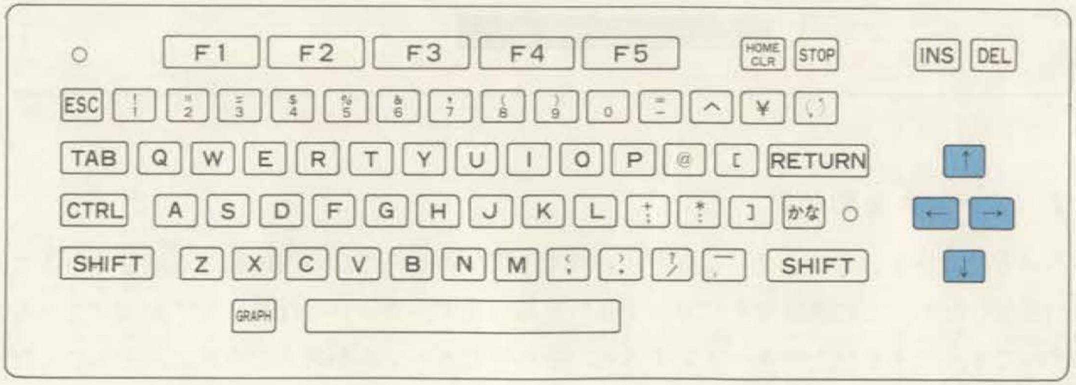


カーソルが画面の右の端まで動いてしまうと，次はその1行下の左の端に移り，またその位置から文字は表示されていきます．このような動作を“自動復帰改行”といいます．

また，文字を画面いっぱいに打ちこんで，カーソルが画面の右下の角まで達したとき，さらにキーインを続けると，突然画面に書いてある文字が1行ずつ上にずれ，いちばん下に1行のすき間ができます．そして，打ちこんだ文字は，そのすき間に表示されることになります．このように，画面が1行くり上がる動作を，“スクロール”といいます．ただしこのとき，画面の下のほうに何やら文字が書かれた長方形が並んでいますが，この部分はスクロールに含まれません．(ここにある文字の表示が，いったい何のためにあるのか，そのわけは第4章で説明します．しばらくそのまま，お待ちください．)

さてこのように普通，カーソルは，自動復帰改行とスクロールの場合を除いて，文字をキーインするにつれ，左から右へ1文字ずつ動くだけです．そして，新しく打ちこんだ文字は，どうしても古い文字の後ろのほうにしか表示することはできません．

しかし，下図の位置にある4個の特殊キーを使うと，カーソルを画面上の自由な位置まで動かし，その位置から新しく文字をキーインして表示させることができます．これら4つのキーのことを，“カーソル移動キー”といい，それぞれのキーに書いてある矢印と同じ方向に，カーソルを動かす働きがあります．



カーソルを動かしていくと，すでに画面上に書いてあった文字の上に，カーソルが重なることがあります．このときカーソルの陰になった文字は，カーソルの点滅につれて，背景の色と文字自身の色が逆転して白地に緑色になった“リバース文字”になったり，普通の状態，すなわち緑色の地に白く書かれた文字になったり，この2つの状態を繰り返します．カーソルが通りすぎれば，元通りになります．

→ のカーソル移動キーでカーソルを動かすとき，画面の右端に達すると，次は自動復帰改行して下の行に移ります．画面の右下の角に達すると，次はスクロールします．つまり → キーの移動のしかたは，一般キーとまったく変わりません．



← のキーを使っていて，画面の左端までカーソルが動いてくると，次は1行上の右端に姿を現します．ちょうど自動復帰改行の逆の動作です．ただし，カーソルがいちばん上の行の左端まで移動すると，もうそれ以上は動きません．この画面の左上の角を，特に“ホームポジション”と呼びます．

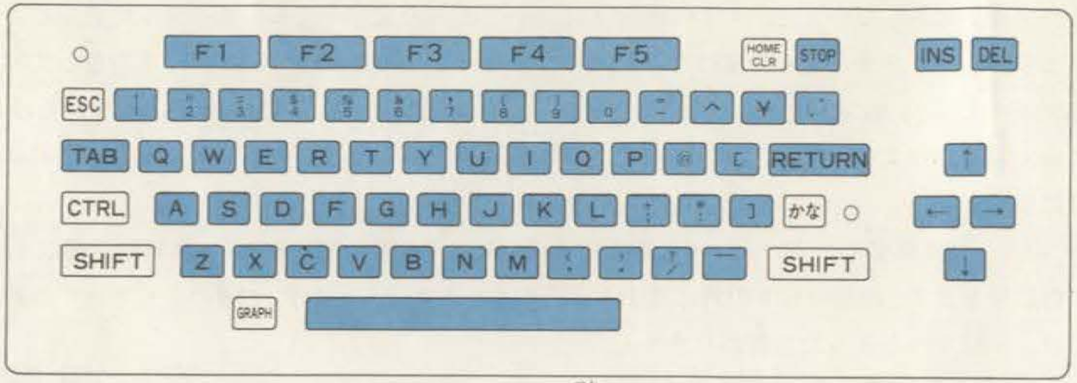
↑ キーや ↓ キーで，カーソルを上下に動かす場合は，どちらも画面の端まで動くと，そこで移動は止まります．

2－3　オートリピート

いままで，キーインしているときに少し長い間キーを押し続けていたら，急にその文字を繰り返し，表示しはじめて，あわてたことはなかったですか．もし1度も経験していなかったら，ぜひとも試してみてください．適当なキーを押して，しばらく（1秒くらい）待っていると，キーは押さえたままなのに，クリック音が“ポッポッポッ…”と鳴り続け，画面には押しているキーの文字が，どんどん表示されるでしょう．



このような動作を“オートリピート”といいます．この機能は，すべての一般キーと，いくつかの特殊キーについて使用できますが，特にカーソル移動キーを使う場合に，とても便利です．他にも，同じキーを何度も続けて押さなければならない場面に出会ったら，積極的にオートリピート機能を使いましょう．



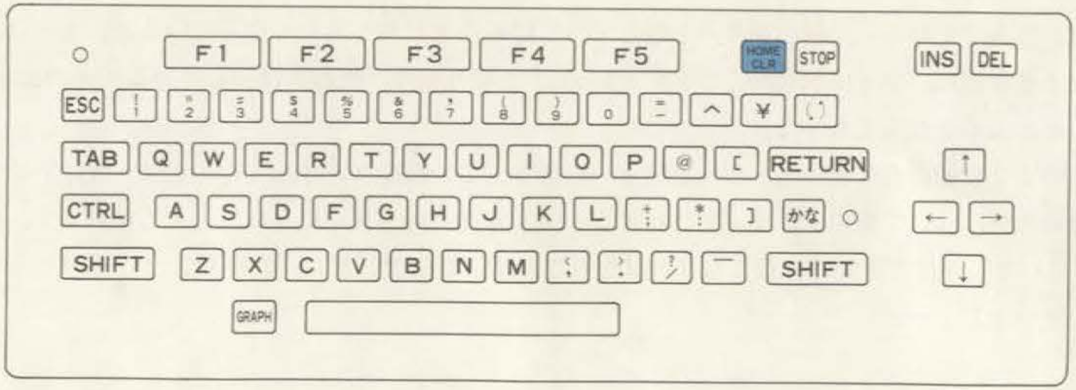
オートリピートの使えるキー

2－4　画面の文字をいっぺんに消す

画面いっぱいに表示されている文字を，すべて消してしまい，気分もあらたにキーボードに向かおうと思ったら，きみならどうしますか．カーソルを画面左上の角，ホームポジションへ移してから，スペースキーを押し続けるとか，[→]を押し続けて画面をスクロールさせ，書いてある文字を画面の外へ押し出す，などの方法を考えつくかもしれませんね．



けれども実は，画面をきれいに消し去るためには，専用のキーが用意されているのです．[HOME CLR] というキーがそれで，"クリアキー"と呼びます．



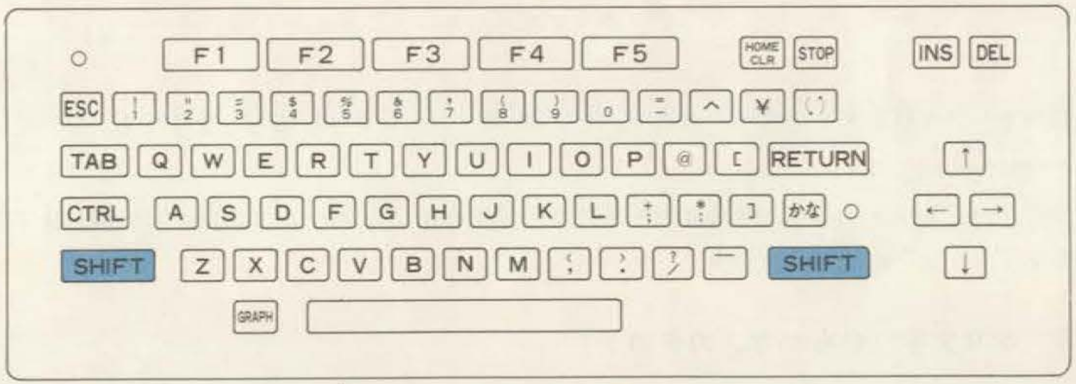
クリアキーを押すと，画面にあったすべての文字は，一瞬のうちに消去され，同時にカーソルはホームポジションへ移動します(ただしこのとき，画面の最下行だけは，やはり消えません).

2－5　シフトキーについて

PCのキーボードには，合計71個のキーが付いています．これはかなり多い数だと感じられるかもしれませんが，実際には，アルファベットの大文字と小文字，ひらがなやカタカナなど，キーインすることができれば便利な文字の数と比べれば，これでもキーの数は，まだまだ足りないのです．

そこで考えられたのが，ひとつのキーを何種類もの文字で共同して使えるような仕組です．下図の位置にある SHIFT というキーは，そのためのキーで，"シフトキー" といいます．

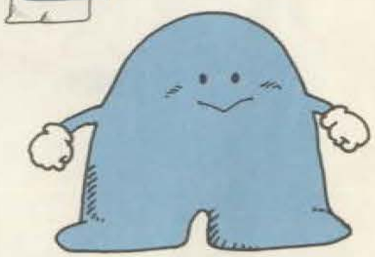






SHIFT を押しながら同時に，A をキーインしてみてください．シフトキーは2つありますが，どちらを選んでもまったく同じ結果が得られます．いまの場合だったら，A のキーに近い左側のシフトキーを使えば良いでしょう．

単に A だけをキーインした場合は，小文字の"a"が表示されるのですが，シフトキーと同時に A をキーインすると，画面に現れるのは大文字の"A"です．シフトキーを押しながらアルファベットのキーを押すと，小文字ではなくて，大文字が表示されるのです．A 以外のキーでも試してみてください．このようにシフトキーを押して，本来そのキーを押したときの文字と異なる文字がキー





インできるようにすることを，“キーをシフトする”といいます．

アルファベット以外のキーをシフトすると，どんな文字が現れるでしょうか．[!/1]のキーをシフトしてみましょう．“！”（エクスクラメーションマーク）が出てきます．キーの上に2個の文字が書いてあるものは，その上側の文字が出てきます．アルファベット以外で，キーに書いてある文字が1個だけのものは，シフトしてキーインしても，何も出てきません．

SHIFT ＋ A ……… “A”　（アルファベットキーをシフトすると，大文字が出る）

SHIFT ＋ !/1 ……… “！”　（2個の文字が書いてあるキーをシフトすると，上の文字が出る）

SHIFT ＋ ¥ ……… “何も出ない”　（アルファベット以外で，1個の文字が書いてあるキーをシフトすると，キーインをしても何も出てこない）

4個のカーソル移動キーとスペースキー，及び[@]キーは，シフトしてもしなくても，キーインしたときの動作は同じです．

クリアキーをシフトすると，画面は消さずにカーソルを左上のホームポジションへ移します．この機能を“ホーム”機能と呼びます．

2－6　かな文字（ひらがな，カタカナ）

PC-6001では，ひらがなやカタカナもキーインできるようになっていて，とても読みやすい文を書くことができます．かな文字を出すためには，[かな]というキーを押してください．するとこのキーのすぐ右にあるインジケータが赤く点灯し，その状態でキーインすると，ひらがなが表示されます．



この[かな]キーを“かなシフトキー”といいます．かなシフトキーは，いったん押すと指を離してもずっと，かな文字が使える状態を保ち続けますが，これを，かなシフトが“ロックされた”といいます．ロックの機能があるおかげで，かな文字を出すときには，アルファベットをシフトして大文字を出すときのように，2つのキーを同時に押す必要はありません．

かなシフトにロックされた状態を元に戻すためには，もう1度，かなシフトキーを押してくださ

い．インジケータが消え，またアルファベットをキーインできるようになります．

ところで，どのキーを押すと，どんなひらがなが出るのか，キーボードにはそのことがまったく書いてありません．この対応関係を知るためには，ひらがなの位置を書いた茶色の板が付属していますから，それをキーボードの上にかぶせて使ってください．この板は“オーバレイシート”，または単に“オーバレイ”といいます．





かなシフトをロックしてキーインすると，オーバレイの，ちょうどそのキーの真下に書いてある文字が現れます．その位置に2つの文字がある場合は，下にあるほうの文字が現れます．上側の文字を出すためには，かなシフトをロックして，さらにシフトキーを押しながら，目的のキーを打ってください．

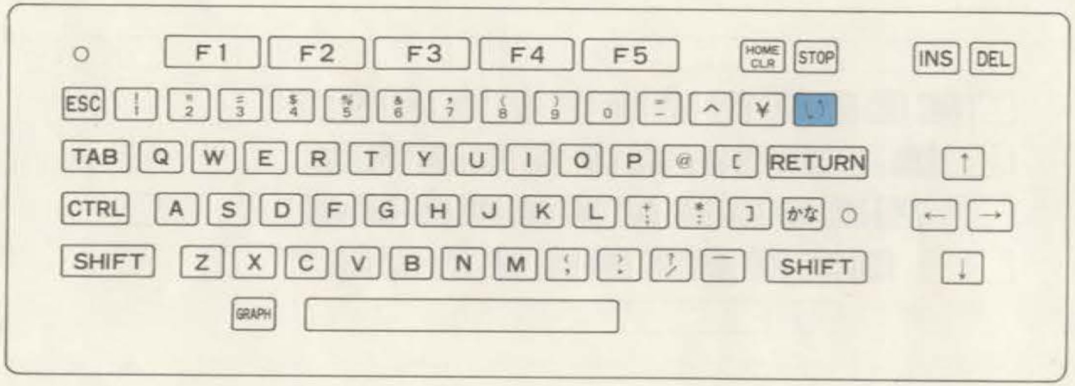
キー	操作	文字
# 3 / あ＼ぁ	本来の状態	“3”
	SHIFT	“#”
	かな	“あ”
	かな ＋ SHIFT	“ぁ”

さて，ここまでの話では，かな文字といってもすべて“ひらがな”ばかりでした．“カタカナ”を

出すときは，次のようにします．

まず，ひらがなの場合と同じように，かなシフトをロックしたら，キーをシフトしながら押してください．こうしてからキーインすると，ひらがなに代わってカタカナが表示されるようになります．もう1度同じようにシフトしながらキーを押せばまた，ひらがな表示に戻ります．このキーのことを“モードチェンジキー”といいます．





2－7　その他の文字や記号

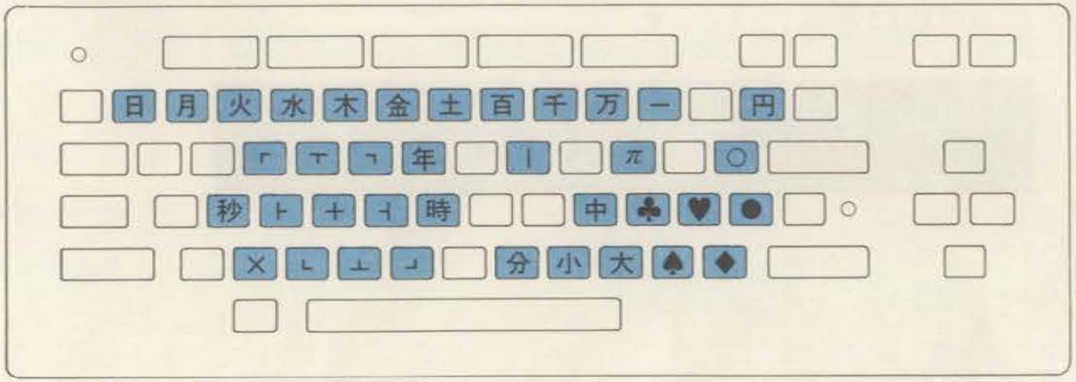


GRAPH のキーは“グラフィックキー”といい，シフトキーと似た機能を持っています．シフトキーを押すのと同じ要領で，このキーを押しながら同時に，何かキーインしてみてください．各キーに対応して，次の図のような文字や記号が表示されます．これらの文字や記号を“グラフィック文字”といい，四角いわく線やトランプのカードなどを簡単に描くには便利なものです．

ただし，キーボードやオーバレイを見ても，グラフィック文字がキーボードにどのように配置されているのかは，書いてありません．グラフィック文字を使用する場合は，右図などを参考にするか，または自分でグラフィック用のオーバレイを作ってしまってもよいですね．

注意してほしいのは，かなシフトを行っているときには，このグラフィックキーはききめがありません．何かのグラフィック文字を出そうとしてキーを押しても，うんともすんともいわない，そ

んな場合はあわてずに，かなシフトがロックされていないかどうか，まず確かめましょう．



2－8　キーインした文字を訂正する

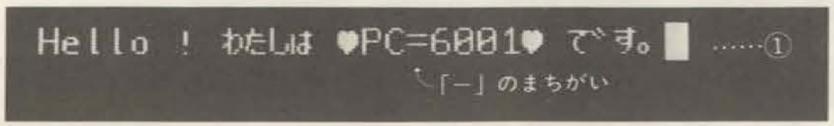


これから，次のような文章をキーボードから打ちこんでみます（キー操作の練習になりますから，きみも実際にキーインしてみてください）．



うまく間違えずに打ちこむことができましたか．

さて，この文章を打ちこんでみたところ，①のようなミスをおかしてしまったらどうしましょう．



このように，打ちこむ文字を取り違えてしまったときには，その場所へカーソルをもっていき，

正しい文字を打ち直せば，それで間違いはすぐに訂正されます．

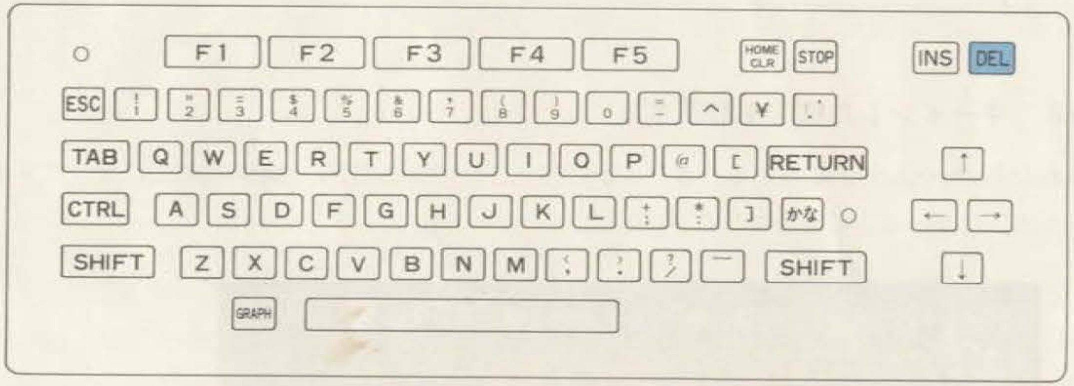
しかし，次の②のように，余分な文字が打ちこまれてしまった場合は，１文字書き直すだけでは間違いは訂正されず，困ってしまいます．

Hello ! わてたしは ♥PC-6001♥ です。█ ……②

余分な文字がはいってしまった



このようなときのための強い味方が，DEL というキーです．このキーは“デリートキー”，または日本語で“削除キー”と呼ばれます．





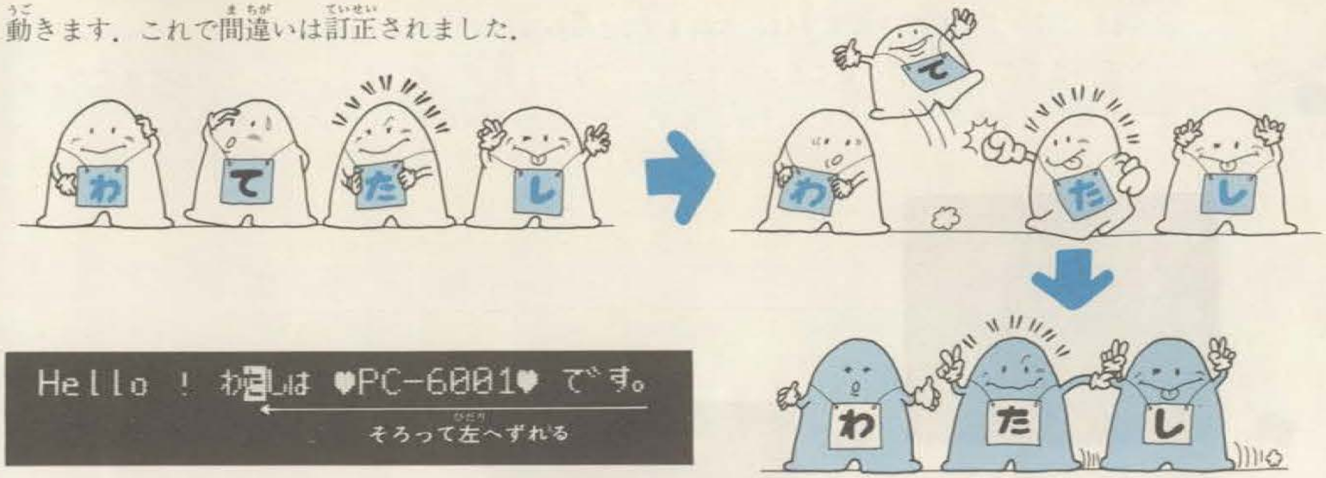
たとえば上の②の場合なら，まず，間違って打ちこんでしまった当の“て”の文字の，ひとつ後ろの文字にカーソルを合わせます．

Hello ! わてたしは ♥PC-6001♥ です。

消したい文字 ↗ ↖ここにカーソルを合わせる

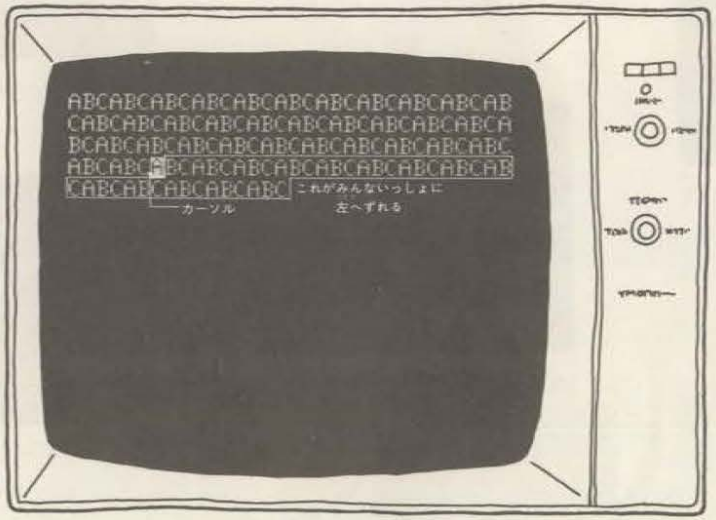
そしてここで，デリートキーを打つと，カーソルの位置の“た”の文字が，カーソルと一緒に左

へ1文字分ずれて移り，それにつれてカーソルの右にある文字も，みんなそろって左へ1文字ずつ動きます．これで間違いは訂正されました．



デリートキーのこの機能は，画面上で何行にもまたがるような長い文章に対しても，使うことができます．ためしに，5～6行続けて文字を打ちこんでおき，カーソルをその中間あたりに持っていって，デリートキーを押してみましょう．

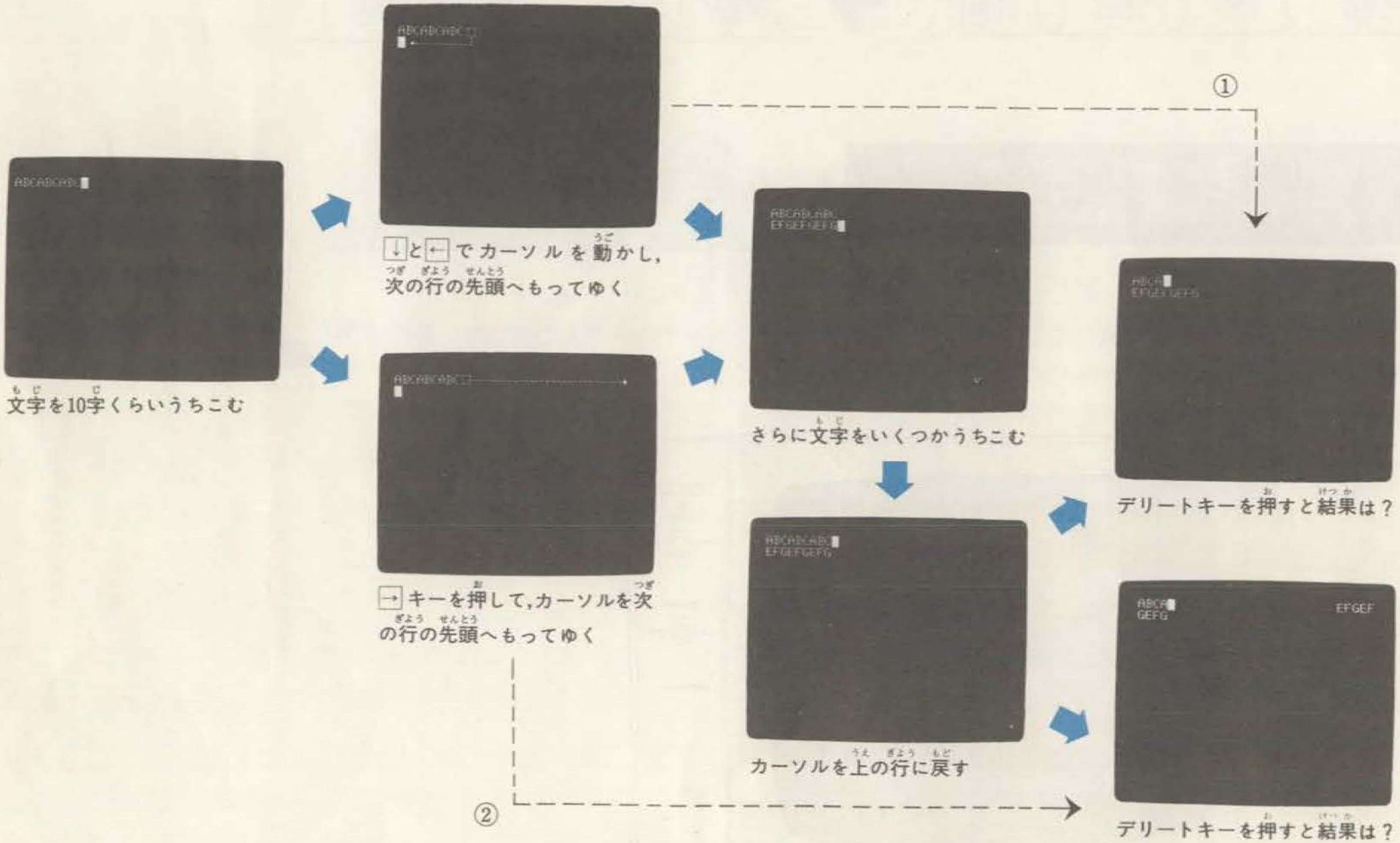




カーソルの後に続く文は，何行あっても，いっしょに引きずられて左へ動いていきます．画面の左端まで引きずられてきた文字は，次は1行上の右端に出てきます．



さて，ここで，問題になることがあります．デリート機能を使うとき，いったいどこまでが，ひとつの文であるとされるのでしょうか．まず，次の2つの例を実行してみてください．

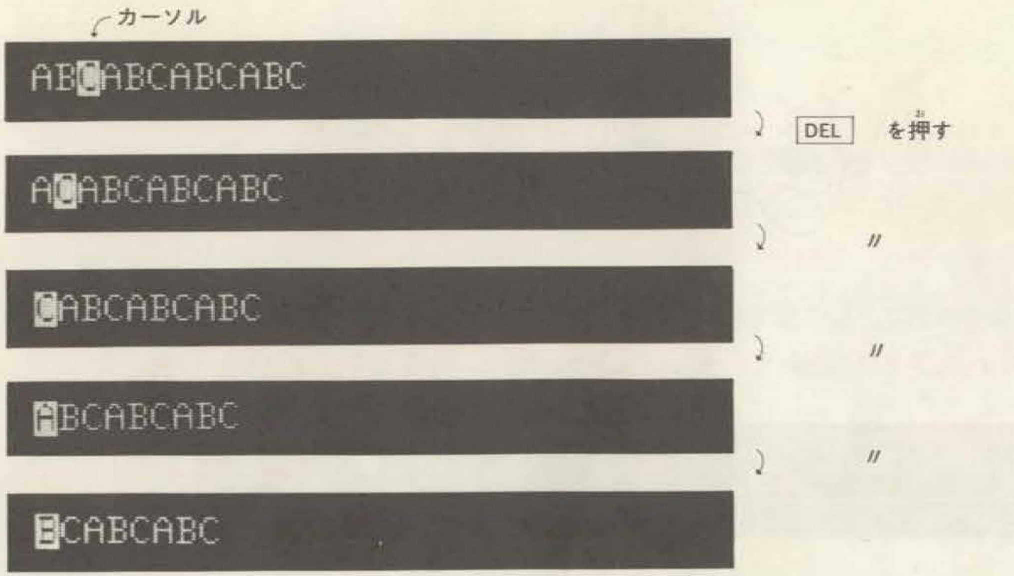


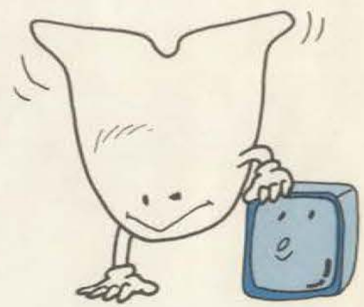
①では，2行書いた文字のうち，上の行だけにデリートキーは作用しますが，②では，下の行もいっしょに引きずって，デリートが実行されます．

これは，カーソルが自動復帰改行すると，そこで2つの行がつながったことになるためです．初めクリアキーで画面をクリアしたときには，画面の16行はすべて独立していて，デリート機能は，その1行の中だけで働きます．しかし，文字を打っていったり，[→] キーでカーソルを動かしたりして，自動復帰改行が起こると，その2つの行は，それ以後はひとつのものとして扱われるようになります．さらにカーソルが動いていけば，自動復帰改行が起こったところは，すべて行がつながってしまいます．デリート機能は，このつながった行の全体，2行分なら32文字，3行分なら48文字，に対して働きます．

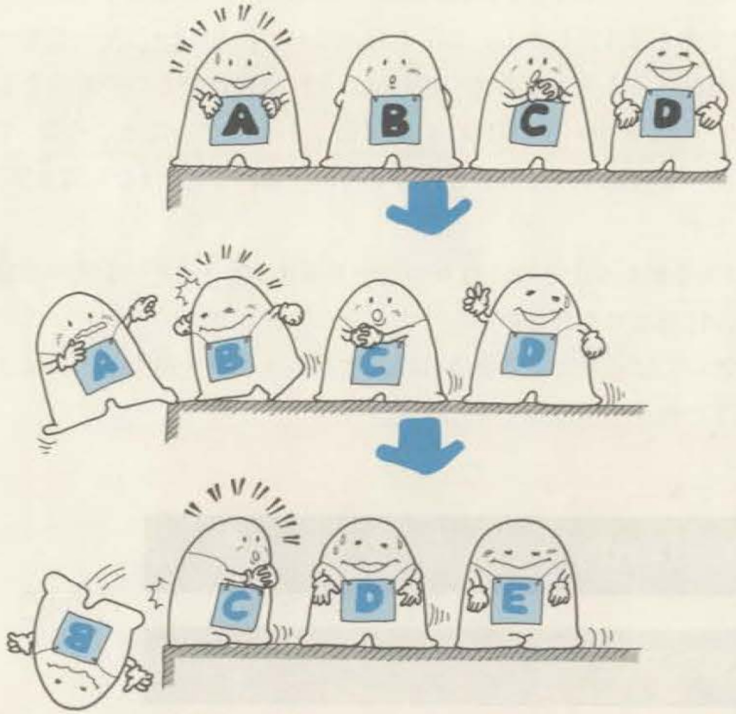
[→] キーでカーソルを動かすときは，行がつながりますが，[←] キーを使った場合には，行は独立したままですから，注意してください．

カーソルが，行のいちばん左にあるときには，デリートキーを押すと，いままでとは変わったことがおこります．下の例をやってみてください．





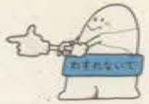
このように，カーソルが行の左端まで達すると，それ以後はデリートキーを押しても，カーソル自身の位置は動きません．そして，カーソルの右にある文字は，右から左へ送られながら，どんどん画面の外に出ていくように消えてしまいます．



ここまでの話で，余分な文字を打ちこんでしまった場合の間違いの直し方は，すべてわかりましたね．次は，文字を打ち忘れてしまったときの，間違い修復法を説明しましょう．

```
Hello ! わしは ♥PC-6001♥ でず。█ ……③
```
"た" の文字を打ち忘れた

こんな場合には，デリート機能とは逆に，文の間に文字を押しこむ機能があるとよいでしょう．そのためのキーが INS キーで，“インサートキー”，または日本語で“挿入キー”という名前です．

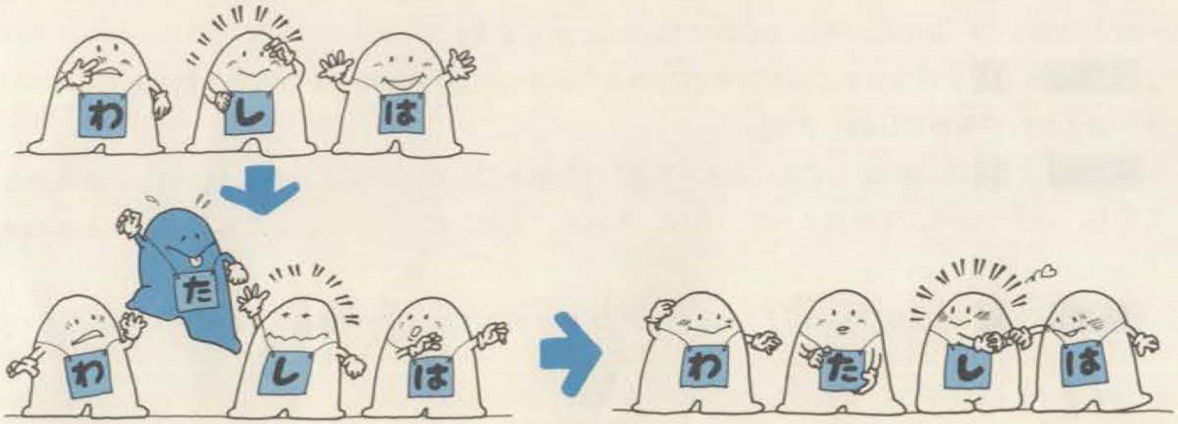


前の③図の場合だったら，まず文字を押しこみたい場所の右側の文字“し”にカーソルを合わせてください．

```
Hello ! わしは ♥PC-6001♥ です。
```

ここでインサートキーを押してみましょう．何も目に見えておこることはないようですね．しかし，次に“た”の文字をキーインすると，インサートキーの効果がわかります．カーソルがひとつ右へ移り，それと共にカーソル以後の文字も，そろって右へ1文字ずつ動くでしょう．そして，カーソルが動いた跡に“た”が表示されます．これで正しい文になりました．

```
Hello ! わたしは ♥PC-6001♥ です。
```
“た”の文字が出る ——————→そろって右へずれる



インサートキーを押しておき，文の最後の文字が画面の右端に達すると，下の行へ押し出されます．もしこの行の下に，何か表示されていた場合，それらはすべて下へ1行ずつずれ，そこでできたすきまにインサートされます．

インサートキーは，かなシフトキーと同じように，1度押すとロックされ，もう1度押すとそのロックがはずれます．またその他に，4つのカーソル移動キーを使っても，やはりインサートキーのロックははずされてしまいます．

2−9　TABキー

TAB キーを押すと，カーソルが右へ動き，画面の左端から8文字目，16文字目，24文字目，32文字目のどれかに達すると，そこで止まります．このキーを“タブキー”といいます．実際に使う場面は，あまりありません．カーソル移動キーの補助として使えばよいでしょう．

2−10　CTRLキー



CTRL キーを，“コントロールキー”といいます．このキーもシフトキーの要領で，他のキーと同時に押して使います．ただし，コントロールキーを使ってキーインする場合，本当に意味があるのは，次の4つです．

CTRL + E は，いまカーソルがある位置から，その行の最後までを消去します．下の行がいまの行と継続しているなら，そこまでいっしょに消去します．

CTRL + U を押すと，こんどはその行の文字がすべて消去されます．そしてそのあと，カーソルは消えた行の先頭に移動します．

CTRL + F は，画面上で次にある“単語”の先頭まで，カーソルを進めます．べつに意味がなくても，いくつかの文字が集まっていれば，それを“単語”だと思って，そこにカーソルを進めます．

CTRL + B は，CTRL + F とは逆に，ひとつ前の“単語”の先頭に，カーソルを戻します．

```
                 ABC DE FGHI
[CTRL] + [B] (               ) [CTRL] + [F]
                 ABC DE FGHI
[CTRL] + [B] (               ) [CTRL] + [F]
                 ABC DE FGHI
```

2－11　モードチェンジキー再び



□のモードチェンジキーは，シフトして押した場合は，ひらがなとカタカナの切り換えを行うものでしたが，このキーをシフトしないで押したら，いったいどうなるでしょうか．

まず画面に適当な文字を書いておいてから，このキーを押してみましょう．画面がクリアされます．それじゃ，クリアキーと同じではないの！　と思うのはまだ早い，もう1度同じ□キーを押してみてください．さっき消えた画面が復活します．

実は，PC-6001は，文字を書いたり消したりするための作業場所をふたつ持っているのです．この作業場所のことを“ページ”といいます．ちょうど，ノートをめくって，あっちのページに書いてみたり，こっちのページに書いてみたりできるようなものです．

テレビの画面には，この2ページのうちの片方が表示されます．最初PCの電源スイッチを入れた時点では，1番目のページが現れていて，以後□キーを押すごとに1ページと2ページが切り換わります．いまの段階では，どちらのページを使っても大差ありませんが，いちおうどちらのページがいま使われているのか知ることもできます．画面の最下行（例の何をやっても消えなかった行です！）の左端に出る数字が，ページの番号を示しています．

2－12　リターンキー

さて，ここでPCにとっては，とても大事なキーについて説明しましょう．

いままでに出てきたキーを打つときは，PCは押されたキーをただそのとおり，ディスプレイ画面に表示するだけでした．右の耳から聞いた言葉が，すぐ左の耳から出ていってしまうようなものです．



きみが本当にPCに何かを伝えるためには，文字を打ちこんだあとに RETURN キーを押さなくてはなりません．このキーを“リターンキー”といいます．リターンキーを押すと，そのときカーソルがある行の文字がPCに取りこまれ，PCはその文字を読んで，それを命令だと思い，実行します．

でたらめな文字をキーボードから打ちこんでおいて，リターンキーを押してみましょう．たいていの場合は，“ピー！”と音がして，

```
?SN Error
Ok
```

と表示されるでしょう．これはPCが，“そんな命令をされてもわかりません”というメッセージを出しているのです．ときには，“ピー”という音が聞こえない場合があるかも知れません．その場合は，ぐうぜんきみの打った文字が，正しい命令だったのでしょう．

どのように文字を打てば，正しい命令となるのか，または間違った命令となるのか，それは第3章から，詳しく説明します．いまのところはただ，PCに命令を伝えるためにはリターンキーを押す必要がある，ということを覚えておいてください．

2－13　キー操作の練習

きみのキーイン能力の力だめしです．次に印刷されているとおりに，キーを打っていってください．正しく打ちこまれたときに限り，楽しいことがおこるでしょう．

```
100 SCREEN 3,2,2:COLOR 2,3 RETURN
110 CONSOLE 0,16,0,0:CLS RETURN
120 CONSOLE 0,6:CLS RETURN
130 DIM A(28),B(28),M$(3),D$(3) RETURN
140 GOSUB 870 RETURN
150 FOR P=48 TO 210 STEP 81 RETURN
160 GOSUB 750 RETURN
170 NEXT RETURN
180 RESTORE 1000 RETURN
```

プログラム続く



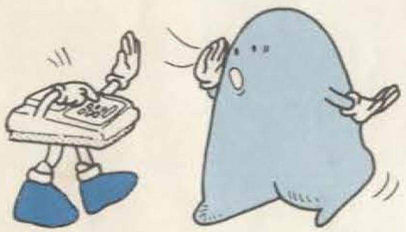
```
190 FOR I=1 TO 12 RETURN
200 GOSUB 610 RETURN
210 NEXT RETURN
220 GOTO 180 RETURN
300 REM************************* RETURN
310 P=K*81-48 RETURN
320 LINE (P,91)-(P+26,105),2,BF RETURN
330 IF D$(K)="" GOTO 400 RETURN
340 LINE (P+4,101)-(P+12,98),3 RETURN
350 LINE -(P+22,101),3 RETURN
360 LINE -(P+12,105),3 RETURN
370 LINE -(P+4,101),3 RETURN
380 PAINT (P+6,101),4,3 RETURN
390 GOTO 420 RETURN
400 LINE (P+4,100)-(P+13,105),3 RETURN
410 LINE -(P+25,100),3 RETURN
420 R1=INT(RND(1)*4)-1 RETURN
430 R2=INT(RND(1)*3)-1 RETURN
440 R3=INT(RND(1)*4)-2 RETURN
450 R4=INT(RND(1)*3)-1 RETURN
460 PSET (P+8+R1,93+R2),3 RETURN
470 PSET (P+20+R3,93+R4),3 RETURN
480 RETURN RETURN
600 REM************************* RETURN
610 READ M$(1),M$(2),M$(3) RETURN
620 READ D$(1),D$(2),D$(3),L RETURN
630 LOCATE 0,5:PRINT:PRINT RETURN
640 FOR K=1 TO 3 RETURN
650 GOSUB 310 RETURN
660 LOCATE K*5-3,5:PRINT D$(K) RETURN
670 NEXT RETURN
680 PLAY M$(1),M$(2),M$(3) RETURN
690 FOR K=1 TO L*150:NEXT RETURN
700 RETURN RETURN
740 REM************************* RETURN
750 FOR I=0 TO 28 RETURN
760 X=A(I)+P:Y=B(I) RETURN
770 IF I=0 THEN PSET(X,Y) RETURN
```



プログラム続く

```
780 IF I>0 THEN LINE -(X,Y) RETURN
790 NEXT RETURN
800 PAINT (P,83),2,2 RETURN
810 LINE (P-11,100)-(P-2,105),3 RETURN
820 LINE -(P+10,100),3 RETURN
830 PSET (P-7,93),3 RETURN
840 PSET (P+5,93),3 RETURN
850 RETURN RETURN
860 REM*********************** RETURN
870 FOR I=0 TO 7 RETURN
880 A(I)=I*5:B(I)=I*I*I/5+80 RETURN
890 NEXT RETURN
900 FOR I=8 TO 13 RETURN
910 A(I)=73-I*5 RETURN
920 B(I)=I*(19-I)+65 RETURN
930 NEXT RETURN
940 A(14)=0:B(14)=140 RETURN
950 FOR I=15 TO 28 RETURN
960 A(I)=-A(28-I) RETURN
970 B(I)=B(28-I) RETURN
980 NEXT RETURN
990 RETURN RETURN
1000 REM*********************** RETURN
1010 DATA c,r,r,ど,  ,  ,1 RETURN
1020 DATA r,e,r,  ,み ,  ,1 RETURN
1030 DATA r,r,g,  ,  ,そ ,1 RETURN
1040 DATA c1,e1,g1,ど,み ,そ ,4 RETURN
1050 DATA c,r,r,ど,  ,  ,1 RETURN
1060 DATA r,f,r,  ,ふぁ,  ,1 RETURN
1070 DATA r,r,a,  ,  ,ら ,1 RETURN
1080 DATA c1o3,f1,a1,ど,ふぁ,ら ,4 RETURN
1090 DATA b,r,r,し ,  ,  ,1 RETURN
1100 DATA r,d,r,  ,れ ,  ,1 RETURN
1110 DATA r,r,g,  ,  ,そ ,1 RETURN
1120 DATA b1o4,e1,g1,し ,れ ,そ ,4 RETURN

run RETURN
```



第3章　BASICを始めよう



きみはもう，第2章を全部読んでくれましたか．キーボードの使い方は，だいたいわかってもらえましたか．あまりおもしろくもない，キーボードの使い方の説明なんていうことに，長い間付き合ってもらい，ごくろうさまでした．けれども，キーボードの使い方をよく知っているのといないのでは，大きな違いですよ．第2章で覚えたことは，きっと役に立つはずです．ほんとに．

さて，この第3章では，いよいよ PC-6001の言葉である“BASIC 言語”を使い，PC をきみの命令どおり働かせてみたいと思います．御期待ください．

3－1　BASICってなんだ？

ここで，1度 PC-6001の電源を切って，もう1度 PC のパワースイッチを ON してください．PC はなりは小さくても，れっきとした“コンピュータ”，つまり“計算機”です．PC ももちろん計算ができます．しかもかなり難しい計算でも，簡単にやってしまいます．たし算なんて，お茶の子さいさい．……と言われて，いま，電卓を使うようなつもりで，

```
3+2=
```

とキーインしました．けれどもこれだけでは，なんにも起こりません．――覚えていますか．PC に何かを伝えたいときには，文字を打ったあとで，リターンキーを押さなければなりませんでしたね．この場合も，忘れずにリターンキーを押してやりましょう．

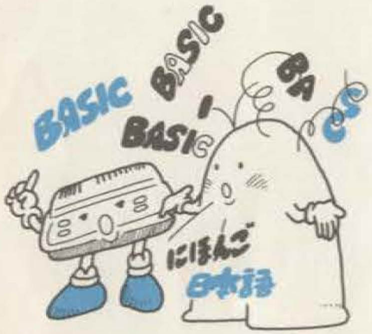
```
3+2= RETURN
```

カーソルが次の行の先頭に移っただけで，答もなにも，まったく出てきません．どうしたことで

しょう．実は，PCが電卓と同じように使えると思ったことが，そもそも間違いなのです．

電卓というのは，計算専用の道具です．だから電卓に，“3＋2＝”と入れれば，いつでもそのとおり計算して答を出します．それ以外の仕事はできませんし，やる必要もありません．

けれどもPC-6001は，ただ計算するだけの機械ではありません．いろいろなことができる可能性を持っています．単に“3たす2だよ”と言われても，それをどうすれば良いのか区別できません．印刷したいのか，テレビ画面に表示するのか，それとも答を記憶しておくのか，いったいどうするんでしょ？　きみがちゃんとPCに，そこまで命令してやらないと，PCは何の仕事もできません．それでは，いったいどのようにすれば，きみが本当にやらせたい命令を，PCに教えられるのでしょう．



相手がもっと簡単な機械，たとえば電気スタンドとか扇風機なら，スイッチをポンと押せば決まった仕事をやってくれます．逆にいえば，その仕事だけしかないのだから，動かしたり止めたりの命令には，スイッチひとつで充分なわけです．しかしコンピュータの場合は，スイッチポン，でどんな仕事もやってのける，という具合にはいきません．なにしろコンピュータにできる仕事というのは，とてつもなく多いのです．それをみんなただのスイッチでコントロールすることは不可能でしょう．

でも，もし命令する相手が人間だったら，どうでしょう．人間は，できる仕事の種類が多いという点では，コンピュータなどより，はるか上を行きます．それなのに，人間にものを指示するときには，スイッチなんて1個も必要ありませんね．それは，“言葉”という強力な手段があるからです．言葉を使って命令するときには，その気になればどんな複雑な仕事でも，細かく詳しいところまでわかってもらうことができます．

実は，コンピュータに命令を与えるときにも，“言葉”を使うのです．ただし，コンピュータには人間の言葉は通じません．人間でも，日本人は日本語をしゃべり，中国人は中国語をしゃべるというように，たくさんの国の，たくさんの言葉があって，自分の国の言葉以外で話されても，よく意味はわかりません．それと同じように，コンピュータにも，コンピュータ自身の言葉というものがあるのです．

PC-6001の場合は，“BASIC言語”という言葉しか理解できません．（もっと詳しくいえば，BASIC

の中でも，特に PC-6001用に作られた“N60-BASIC 言語”というものだけです.）きみが PC に命令を与えるときには，この BASIC という言葉を使って話しかけてやって，はじめてその命令を受け入れてくれるのです．

というわけで，これからきみに，BASIC 言語の使い方を覚えていってもらいます．そんなに心配しなくてもいいですよ．BASIC という言葉は，英語なんかよりずっとやさしいんですから．

3－2　計算すると答えはいくつ？

それではさっそく PC に，“3たす2”を計算して画面に表示するよう，命令してみましょう．BASIC では，この命令を次のように表現します．

```
? 3+2    ……①
```



最初の“?”（クエスチョンマーク）が，その主役です．この“?”の意味を翻訳すれば，“うしろにある式を計算して，その答を画面に表示しなさい.”なんていう感じですね. この場合なら，“3＋2”を計算して，答を表示してくれるというわけです．そうそう，このような命令を実際に PC に送りこむためには，何度もいうようですが，くれぐれも最後に RETURN を押すのを忘れないでください．



①を RETURN で実行させると，画面には，

```
5
Ok
```

と，みごとに答が表示されました．BASIC の場合，電卓のように，計算の最後に“＝”をつける必要はありません．なんだったら“＝”をつけて，試してみましょうか．

```
? 3+2=   RETURN
```

これを実行すると，“ピー！”という音がして，

```
?MO Error
Ok
```

と画面に表示されます．

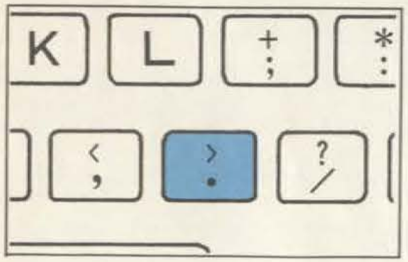
BASICという言語では，ちょっとした間違いがあってもいけません．このように，最後にひとつ“＝”という余計な文字をつけただけで，PCはもう許してくれず，音や文字できみに注意を与えたのです．このように言葉使いに厳しいというのは，なんだか不便なようにも思えますが，いつも正確な動作をする，というためにはしかたがないことかもしれません．

たとえば人間にものをたのむときには，きっちりとした言葉使いでなくても，だいたい適当にわかってくれるのですが，そのために大変な間違いをする場合もでてきます．「おーい風呂をたのむ」と言われて，お風呂の栓を抜いたのに，実はお風呂をわかし直して欲しかったんだ．なんてことはまあ許せるかもしれませんが，その逆だったら火事になってしまったかもしれません．

さてもう1度，計算の方法に話を戻しましょう．次の例を実行してみてください．




```
? 9-3 [RETURN]                ……②
? 1+2+3+4+5 [RETURN]          ……③
? 8 [RETURN]                  ……④
? 2.7-3.14 [RETURN]           ……⑤
? 1 2  +  3 4 [RETURN]        ……⑥
```



みんな正しい答を表示しましたね．③のように，多くの数の間の計算をまとめて実行できるというのも，電卓に勝るところです．④のように，数がひとつポツンとあれば，その数をそのまま表示します．

小数点記号を使うときは，⑤にあるように，“.”（ピリオド）を使ってください．このキーは“,”（カンマ）とすぐ隣りあっているので，よく間違えることがあります．注意しましょう．（ピリオドとカンマの押し間違いで，コンピュータが狂い，ロケット打ち上げに失敗したという伝説もあるくらいです．）

また，⑥のように，間にいくつの空白をはさんでも，答には関係しません．つまり，この⑥は，

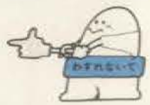
“?12+34” と同じことです.

3－3　かけ算とわり算と

たし算，ひき算だけではつまらないし，なんだか簡単すぎて，頭の良いPCに申し訳ないですねえ．ここはひとつ，かけ算やわり算にも挑戦してみましょう．……とはいうものの，かけ算記号の“×”や，わり算記号の“÷”はどこにあるのでしょう．キーボードにはみあたりません．

実は，コンピュータの世界では，かけ算とわり算の記号は，普通の算数や数学の教科書とは違うものを使います．それは次の2つの記号です．

かけ算――“*”（アスタリスク）
わり算――“／”（スラッシュ）



では，この記号を使ってさっそく練習してみましょう．次の例を実行してみてください．なお，以後，RETURN は省略しますので，命令の最後には必ず RETURN をキーインしてください．

```
? 7*3                      ……①
? 5/8                      ……②
? 6/5*4/3*2/1              ……③
? 2*2*2*2*2*2*2*2*2*2      ……④
```



正しい答が出ましたか？　要するに，記号の形が変わっただけで，やっていることは普通のとおりの，かけ算やわり算です．

②の答は，“.625”となっていますが，これは“0.625”のことです．PCでは，1より小さい小数，つまり“0.……”というように“0”で始まる小数を表示する場合，その先頭にある“0”を省略してしまい，小数点“.”から後だけを画面に出します．きみがキーボードから数字を打ちこむときには，この先頭の“0”は，つけてもつけなくてもかまいません．どちらでもお好きなように．

さて上の例で，④をキーインするのは大変だったでしょう．同じ“2”が何個もつながっていて，いまどこまで打ちこんだんだっけ？　なんてことになりませんでしたか．実はこういうように，同じ数を何度も何度もかけあわせるときには，もっと気のきいた方法があります．そのための記号が

“∧”（アップアロー）で，次のように使います．




```
? 2^10 ……⑤
```

この“2∧10”は，“2の10乗”と読み，たったこれだけで，2を10回かけあわせたのと同じ結果の1024が得られるのです．この“∧”を使った計算を，**“べき乗計算”**といいます．なお，このべき乗計算の書き表しかたも，実は通常のものとは異なっていて普通の教科書などでは，2の10乗のことを“2^{10}”と表現しています．

もちろん，2や10以外の数でもべき乗の計算はできます．次の例を試してみてください．

```
? 5^3     ……⑥
? 4^7     ……⑦
? 2^3^4   ……⑧
? 2^-3    ……⑨
```

⑨は，どういう意味かわかりますか．2をマイナス3回かけ合わせる？……と考えると，何のことやらわかりません．これは次のように考えてください．2∧3は8，2∧2は4，2∧1は2です．ということは，

```
2^3 --> 8
           ) 2で割る
2^2 --> 4
           ) 2で割る
2^1 --> 2
```

順に2で割った値となっているのがわかるでしょう．この方向に話を進めていけば，2∧0は1．2∧−1は0.5……という答が出てくるのが自然の理というものです．

```
2^1  --> 2
              ) 2で割る
2^0  --> 1
              ) 2で割る
2^-1 --> .5
              ) 2で割る
2^-2 --> .25
              ) 2で割る
2^-3 --> .125
```

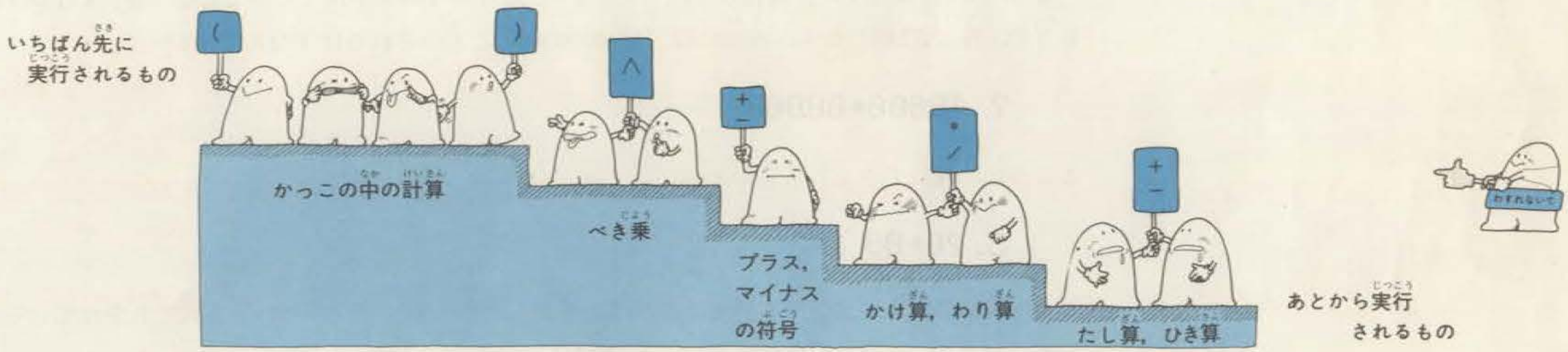
ちょうど，2のマイナスx乗は，2のx乗分の1に等しいという関係になっています．これは2以外の数でも同じです．いくつか試して，確認してみましょう．

```
? 2^-4   ……⑩ ) 等しい
? 1/2^4  ……⑪
? 4^-3   ……⑫ ) 等しい
? 1/4^3  ……⑬
? 5^-2   ……⑭ ) 等しい
? 1/5^2  ……⑮
```

3－4　何をどんな順番で

計算をするとき，忘れてならないのが，計算の順序です．同じひとつの計算の中に，たし算とかけ算があれば，かけ算を先に行います．べき乗計算は，かけ算よりもさらに優先的に実行されます．これは，BASICだけの特例ではなく，どこへいっても通用する共通の計算規則です．

その他のわり算，ひき算などについても，PCが計算する順序は決まっていて，まとめると次のようになります．



かけ算とわり算は，優先される順番が同じですから，並んでいる順に左から実行されます．たし算とひき算がいっしょにある場合も，同様に左のほうから順に実行されます．

それでは，次にいくつか例をあげてみます．答はいくつになるでしょうか．最初に自分で考えて

みて，そのあとでPCに正解を教えてもらってください．



```
? 6+5 * 4-8 /2    ……①
? (6+5)*(4-8)/2   ……②
? 3/4*3/4         ……③
? (3/4)^2         ……④
? 3/4 ^2          ……⑤
? -3 * -3         ……⑥
? (-3)^2          ……⑦
? -3 ^2           ……⑧
? 4^ 3^2          ……⑨
? 4^(3^2)         ……⑩
```

このような計算の優先順位は，知っているに越したことはありませんが，複雑な計算式では，あまりこの知識にたよらないほうがりこうです．先に計算させたいところは，積極的にかっこでくくってしまいましょう．あとあと自分で見なおすときにも，そのほうが楽ですよ．

3－5　大きな数や小さな数

まず，次の計算をPCで実行してみてください．どういう結果が出てくるでしょうか．4万かける8万で，答は32億だから，画面には“3200000000”と表示されるはずなんですが……．

```
? 40000*80000    ……①
```

ところが，現れてきた答を見ると，

```
3.2E+09
```

という，見慣れない文字です．実はこれは，答が“浮動小数点表示”という方式で表されているのです．

世の中には，とても大きな数が存在します．銀河系のさしわたし（1兆キロメートルの100万倍）とか，地球の重さ（1兆トンの60億倍）など，もう“億”や“兆”の単位をはるかに超えていますね．こういう数を，まともに数字で表そうとすると，めったやたらに桁数ばかり多くなってし

まいます.

銀河系は——1000000000000000000 キロメートル

地球は———6000000000000000000000 トン

これでは読むのも書くのもひと苦労，そこで考えられたのが浮動小数点表示です．これは，10のべき乗を使って大きな数を表現しようというもので，たとえば“1兆の10億倍”といった数も，べき乗の形にすれば，10の22乗（10∧22）にしかなりません．この“10∧22”を単位として考えれば地球の重さはちょうどその6倍，つまり“6＊（10∧22）”ということになります．ただしこのまま書いたのでは，計算式と区別できませんから，これを“6E22”と書くことにします．これが，浮動小数点表示です．このとき“E”の前にある数を“仮数”，“E”の後にある数を“指数”といいます．





N60-BASICでは，そのまま画面に表示すると10桁以上になるような数は，浮動小数点表示にしてしまいます．次の例で確かめてみましょう．

```
? 999999999        ……②
? 999999999+1      ……③
? 11409*10821      ……④
? 21642*57045      ……⑤
? 10^8             ……⑥
? 10^9             ……⑦
```

ところで，PCはこのように，浮動小数点表示を使うか使わないかの境目を，10桁以上か9桁以

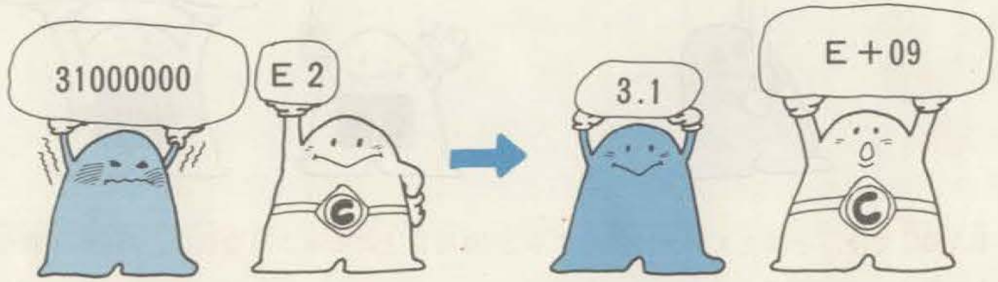
下で分けていますが，きみが浮動小数点表示で数を書くときまで，それに合わせる必要はありません．10桁以上の数をそのまま書いても，9桁以下の数を浮動小数点表示で書いても，どちらでもかまいません．その他，浮動小数点表示では下の例のような使い方をしても大丈夫，PCはちゃんと理解してくれます．



```
? 1E2              ……⑧
? 7000000000000    ……⑨
? 10000000E2       ……⑩
? 2.7e4            ……⑪
? 3 2 e 1 0        ……⑫
```

⑧は桁数の少ない数を浮動小数点表示で表した例，⑨は逆に13桁もある数をそのまま書き表した例で，どちらもPCは受け入れて，その証拠にPC自身の表示法に直して画面に表示してくれます．

⑩のように，仮数部分が10以上の場合，PCは自動的に，仮数の整数部分が1桁になるように直します．これを“正規化”といいます．



⑪，⑫は，大文字の“E”ではなくて，小文字の“e”でも良いという例です．とはいえ⑫のような書き方では，読みにくいことこのうえないですから，なるべくなら大文字を使ってください．

次にもうひとつ，浮動小数点表示を使うと小さな数の表示も便利になります．この場合は，10のマイナスのべき乗を使うのです．たとえば，“100万分の1”という数は，“1/ (10∧6)”ですから，これは“10∧－6”と同じことですね．

PCでは，0.01より小さい数を表示しようとすると，浮動小数点表示に直します．とにかくまた

いくつかの例をあげてみますから，これを実行して感じをつかんでください．




```
? 1/100        ……⑧
? 1/1000       ……⑨
? (1/10)^4     ……⑩
? 0.01-0.001 ……⑪
? 0.01+0.001 ……⑫
```

3－6　どこまで数字は正確か

きみが紙の上で計算する場合でも，2桁や3桁の数を計算するよりも，6桁も7桁もある大きな数を計算するほうが，時間はかかるし面倒だし，いやだなあと思うでしょう．コンピュータがいくら計算が速いからといっても，やはり何十桁もの数の計算をさせていたら，それなりに時間がかかってしまい大変なのです．そこでPC-6001では，数を扱うときには，上から9桁目までを使用し，それ以下は四捨五入してしまいます．




```
? 100000000.4 ……①
? 100000000.5 ……②
```

四捨五入なので，①は右端の“4”は切り捨てられ，②の右端の“5”は切り上げられるというわけです．

えっ，たった9桁しか計算しないで，あとはうやむやだなんて，そんないいかげんなことで良いのか！などと思わないでください．9桁まで正しいということは，大変なことですよ．これは，重さ1千トンの鉄の球に，1円玉が乗っかっているかいないかを区別しているのと同じくらいのことをしているのですから．

以上のようなわけで，PCに数を与えるときに，何桁も細かいところまで書いても，あまり意味はありません．このように，どこまで正確な数として受け入れてくれるか，ということを，コンピュータの“有効桁数”といいます．PCの有効桁数は9桁であるというわけです．

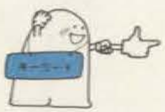


3－7　大きすぎる数，小さすぎる数

浮動小数点表示なら，どんな大きい数でも，どんな小さな数でも，自由に扱えるかといいますと，残念ながらそうではありません．次の計算を実行してみてください．

```
? 10^-40 ……①
? 10^40  ……②
```

本当ならば①は，“1E-40”という答が返ってくるはずです．それなのに，画面に現れてくるのはたったの1文字の“0”．②の場合はもっと悪い．“1E+40”とは表示されずに，“ピー！”という音と共に，“?OV Error”の文字が出てきて，PCにしかられてしまいました．



PC-6001のBASICで使えるのは，だいたい3E-38から1.7E+38までの範囲の数に限られます．3E-38より小さい数は，PCは0だと思ってしまいます．1.7E+38より大きい数を使おうとすると，PCは“そんなに大きい数は困ります”というメッセージ“?OV Error”を出して止まってしまいます．でもまあ，実際にはこんな極端に大きい数や小さい数を使うことはありませんから，心配はいらないでしょう．

3－8　1度にいくつでも答えます

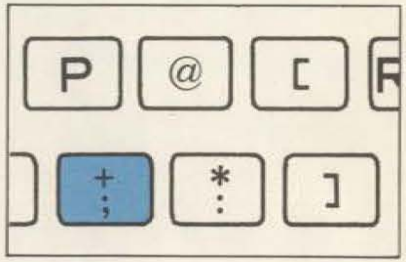
さてここで，カンマ(,)とセミコロン(;)を紹介します．これらを使うと，いくつもの計算式の答を1度に表示させることができるようになるのです．

まず初めにカンマを使った例を見てもらいましょう．次のとおりにキーインし，実行させてみてください．

```
? 1+2,3-4,5*6,7/8  ……①
 3             -1
 30             .875
Ok
```

このとおり，カンマで区切った4つの式はそれぞれちゃんと計算されて，横1行に2つずつ答が表示されました（4回別々に計算するより，ずっと簡単になったでしょう）．

セミコロン（；）を使っても，同じようにいくつもの式を1度に計算できますが，表示するときの形が違います．①のカンマをセミコロンに変えた次の例を実行して結果を比べてみてください（セミコロンと右隣りにあるコロンキーと間違えないように）．




```
? 1+2;3-4;5*6;7/8 ……②
 3-1 30 .875
Ok
```

こんどは答の間の距離が，さっきと比べてぐっと詰まって表示されましたね．このようにセミコロンで区切ったときには，すべての答はぴったりとくっついて表示されます（ただし0以上の数ではプラス符号の位置に何も表示しないので，実際には1文字分のすきまがあるように見えます）．

これに対してカンマで区切った場合は，1行の左端と，左端から15文字目の位置に数が表示されます（このときも，0以上の数はプラス符号を表示しないために，そのひとつ右の位置から書かれているように見えます）．

カンマを続けて書くと，何も表示はしないで次の位置までカーソル（目に見えないカーソルがあるものと考えてください）が動きます．セミコロンはいくつ続けても何も起こりません．次の例で確かめてください．

```
? 1,,2,3,,4       ……③
 1
 2              3
                4
Ok
```




```
? 1;;2;3;;4       ……④
 1 2 3 4
Ok
```



カンマとセミコロンを混ぜて使うこともできます．

```
? 1,,2;3,,4;5;6 ……⑤
 1
 2 3
 4 5 6
```

カンマを使った場合でもセミコロンを使った場合でも，1 行がいっぱいになれば無理をしないで次の行へ移り，そこから表示を始めます．

```
?98765*43210,12345 ……③
 4.26763565E+09
 12345
```

この③の例では，1 番目の式の答だけでもう画面の 15 文字目まで使ってしまったため，次の数は一段下の行に表示されています．

```
? 12^34;123456;12/34 ……④
 4.92223552E+36 123456
 .352941177
```

この④の例では，2 番目の答を表示した後，画面の右端に残っている 10 文字分の空間に 3 番目の答はもう入りきらないことを PC が自動的に判断し，次の行から表示を始めています．

```
? 12^34;12345;12+34  ……⑤
 4.92223552E+36 12345 46
```

これは④の 3 番目の計算がわり算からたし算に変わっただけですが，こんどは 3 つの答が充分 1 行に入りきる，と PC は判断し，そのように表示しているわけです．



3－9　RETURNキーの力の限界

リターンキー RETURN を押すと，PC は画面にある文字を実際の命令として読みこみます．このことについて，いままでは何の問題もありませんでした．しかし，カンマやセミコロンを使って 1 行が長くなってくると話が違ってきます．たとえば次の命令をキーインし，実行させてみてください．

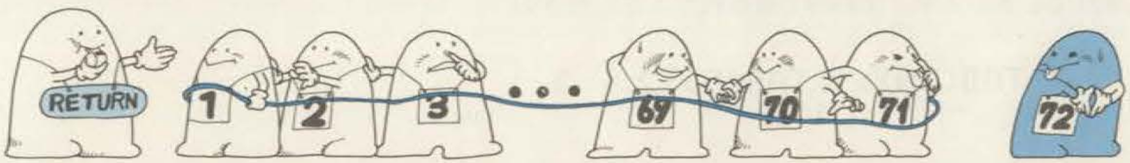
```
? 1111111,2222222,3333333,444444
4,5555555,6666666,7777777,888888
8,9999999
```

結果は次のようになります．

```
1111111        2222222
3333333        4444444
5555555        6666666
7777777        8888888
99999
```

最後だけ“99999”と5桁になってしまいました．これは，その後ろにある2文字が読みこまれなかったためです．どんなに長い命令をキーインしても，リターンキーで実際に読みこまれるのはその先頭から71文字まで(つまり2行と7文字まで)でしかありません．だから命令は長くても2行以内になるべくおさまるよう，いつも気をつけていてください．





3－10　表示の位置はお望みのまま

カンマとセミコロンだけでは，1行に2つの数の割合で表示するか，数と数の間をベタ詰めにして表示するか，この2つだけしかできません．そこで威力を見せるのが，これから説明する2つの機能，“SPC関数”と“TAB関数”です．



SPC関数を使うと，表示したい数の間に好きなだけのスペースをあけることができます．次の例を実行してみてください．

```
? 1;SPC(5);2;SPC(17);3          ……①
 1      2                  3
```

←5文字→　←――――17文字――――→

単にセミコロンで区切ったときに比べて，SPC（5）では5文字分，SPC（17）では17文字分だけ，数と数との間隔が多くとられています．つまりSPC(x)と書くと，x文字分のスペースをはさむことができるわけです．

TAB関数の場合は，数と数の間にスペースをとるのではなく，行の左端から何文字目に表示するのかを指定できます．試してみましょう．

```
? 1;TAB(5);2;TAB(17);3 ……②
 1    2           3
←5文字→
←――――17文字――――→
```

TAB（x）と書くと，そのときカーソルがある位置から，かっこの中のxで指定される位置までスペースが入ります．結局次に表示される数は行の左端からx＋1文字目に現れてくるわけです．

もしもそのときカーソルがある位置よりも左の位置をTAB関数で指定した場合には，それは無視されてしまいます．次の例で確かめてみてください．

```
? TAB(10);1;TAB(5);2 ……③
           1 2
```



ここで“関数”という言葉について少し説明しておきます．SPC（x）やTAB（x）のように，ある数xが変わるとそれについて得られる結果が変わってくるようなもののことを，一般に“関数”といい，変化の元となるxのことを“引数”といいます．引数には，ただの数でなくて計算式を使うこともできます．その場合は，式を計算した答が実際には使われることになります．たとえば次の例を試してみてください．

```
? TAB(3+4);1;TAB(4*6);2 ……④
        1                2
```

引数はその大きさがある範囲に限られていることがあります．TAB関数とSPC関数の場合は0から255までの間に引数の値が入らなければいけません．

```
? TAB(300);1 ……⑤
? TAB(-1);2  ……⑥
```

⑤や⑥のように，引数が許されている範囲からはずれると，“?FC Error”というメッセージが現れます．これは関数の使い方を間違えると出てくるもので，これから後もよくお目にかかることになると思います．



3－11　関数のいろいろ

関数といえば，中学や高校で誰でも1度は頭を悩ませる三角関数などが思い出されますが，PCはこれらの関数を計算することも得意です．その他どんな関数があるのか，次にまとめてみました．

① SIN (X)　　正弦関数
② COS (X)　　余弦関数
③ TAN (X)　　正接関数



これら3種の三角関数は，ラジアン単位で引数を与えなくてはなりません．30°とか45°のように普通の“度”単位で表された角度は，ラジアン単位に直してから使う必要があります．その方法は，次のとおりです．

ラジアン＝度＊3.141592652／180

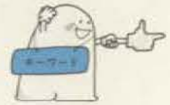
たとえばsin 30°を計算したければこうします．

```
? SIN(30*3.141592652/180)
```

④ LOG (X)　　対数関数
⑤ EXP (X)　　指数関数



対数関数は，Xの自然対数つまりeを底とするXの対数を求めます．Xは0以下の数ではいけま



せん．指数関数はeのX乗を求めます．

⑥ SQR (X) 平方根

普通はXの平方根を$\sqrt{X}$と表しますが，PCのBASICはこのようにSQR (X) と書きます．引数は負の数ではいけません．

⑦X∧(1／Y) べき乗根

これはちょっと関数とはいえないかもしれませんが，ここで紹介しておきます．これでXのY乗根を求められます．Xは負の数ではいけません．

ここまでは，数学の教科書に出てくるような関数ばかりでしたが，他にもおもしろい関数がいくつかあります．続けて説明していきましょう．



⑧ ABS (X) 絶対値

普通はXの絶対値を |X| と表しますが，BASIC ではこのように書きます．



⑨SGN (X) 符号関数

Xの符号を判断します．Xが正ならSGN(X)は1，Xが0ならSGN(X)も0，Xが負ならSGN (X) は－1になります．次に例を示したので見てください．

```
? SGN(-2);SGN(3);SGN(0)
-1 1 0
```



⑩ INT (X) 整数化関数

X以下でいちばんXに近い整数を得る関数です．Xが正の数の場合は小数点以下を切り捨てることと同じです．次の例を見てください．

```
? INT(1);INT(1.5);INT(2)
 1 1 2
Ok
```

```
? INT(-2);INT(-1.5);INT(-1)
-2-2-1
```

このINT 関数の応用を2つだけあげておきましょう.

⑪INT (X+0.5)　　小数点以下四捨五入

ただしXは正の数でないといけません.

⑫X−INT (X/Y) ＊Y　　剰余

XをYで割った余を求めます. たとえば次の例を見てください.

```
? 100-INT(100/7)*7
 2
```

100わる7の余は, 確かに2ですね.

3−12 いつまでも覚えていたい

PCはここまで見てきたように, いろいろな計算をすることができます. けれどもせっかく画面に表示された答も, いくつかの計算を続けているうちに, スクロールして画面の外へ消えてしまいます. これはもったいない, なんとか記録しておきたいものですね. きみならどうしますか. 紙に書いてとっておきますか？ なに, 暗記には自信があるから自分で覚えておくって？ いやいや, そんな面倒なことをする必要はありません. PCが出した答はPC自身に覚えてもらえばよいのです.

たとえば6001という数を覚えさせてみましょう. それには次のようにキーインして実行してください.

```
m=6001
```
……①



これでオーケー, PCは6001を記憶しました. 本当に覚えたのかどうか調べてみましょうか.

```
? m
 6001
```

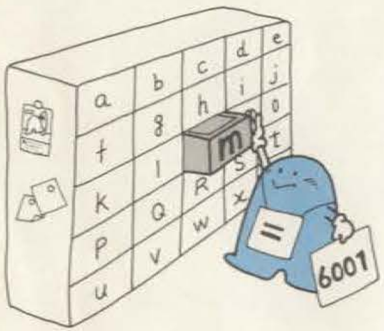
どうでした．ちゃんと覚えていましたね．ここで出てきた“m”は，電卓でいう“メモリ”のようなもので，PC ではこれを“変数”といいます．そして，この“変数”に数を覚えさせることを，“代入する”という言葉で表します．

昔の BASIC 言語では，変数に何かの数を代入するときには初めに“LET”という文字をつけて，

```
LET m=6001
```

のように書かなければなりませんでした（そのため今でも，この代入命令のことを“LET 命令”と呼ぶことがあります）．しかしこの N-60BASIC では，“LET”を省略しても，まったくかまわないのは見てのとおりです．





変数 m に
6001をしまいます．

さて，いま変数 m に代入されている数は 6001 ですね．ここで次の代入命令を実行すると，m の値はどう変わるでしょうか．

```
m=123
```

それでは m の内容を見てみましょう．

```
? m
```

m は 123 に変わってしまいました．このように新しく代入命令を実行すると，古い内容はすっかり消えてしまうのです．これでは覚えていられる数というのは，最後に代入したもの 1 個だけしかないことになりますね．

しかし，変数は“m”ひとつだけではありません．実は BASIC には何百個もの変数が用意されていて，それぞれに違う数を代入することができるのです．それらの変数は区別のため，ひとつひとつ名前がついています．“m”というのも，そのうちのひとつです．

変数名は，まずアルファベット（a~z）で始まっていなくてはなりません．いちばん簡単なのは，アルファベット 1 文字という名前です．

```
a=256
x=192
? a,x
```

名前の2文字目以後は，アルファベットの他に数字が使えます．

```
pc=12345
basic=777
v100=4
Im4u=1000
? pc,basic,v100,Im4u
```

アルファベットや数字以外の文字を変数名に使うとエラーになります．さらにもうひとつ，すでにBASICが使用している名前は使えません．これはその名前が変数名の一部に含まれているだけで，エラーになります．次にあげるのは，すべてエラーになる例です．

```
3p=10       ←(先頭がアルファベットでない)
a#=5        ←("#"はアルファベットでも数字でもない)
sansin=7    ←("Sin"が含まれる)
```

こんどは大文字と小文字の違いを調べてみましょう．次のとおりに打ちこんで実行してみてください．

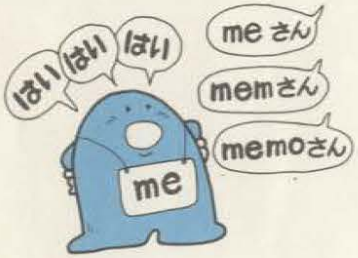
```
a=1
A=2
? a,A
```

両方とも答は2になってしまいましたね．実は変数名は，大文字で書こうが小文字で書こうが，同じアルファベットであれば同じ名前だとされるのです．つまり，上の例では，aもAも同じ変数を表していたわけで，結局次のように実行したのと変わらないのです．両方とも2になったのはあたりまえでした．




```
a=1
a=2
? a,a
```

それではこんどは次の命令を順に実行してみてください．どうなるでしょう．




```
m=1
me=2
mem=3
memo=4
? m;me;mem;memo
```

あれ！　2番目の“me”と3番目の“mem”は，どちらも2や3ではなく4に変わっていますね．このBASICでは，変数の名前は，最初の2文字だけしか見ていないのです．もし最初の2文字が等しければ，その2つの変数は同じものだとされてしまいます．

3−13　変数の便利な使い方

たくさんの変数を代入するとき，ひとつの代入命令ごとに，いちいちリターンキーを押すのはけっこう面倒なものです．そういうときには次のような方法が使えます．

```
a=1:b=2:c=3:? a,b,c ……①
```



このように“コロン（：）”を使っていくつかの命令をつないだものを“マルチステートメント”といいます．こうすると，最後に1回リターンキーを押すだけで，すべての命令が左から順に実行されます．これは代入命令だけに限らず，これから出てくるどんなBASICの命令でも同じです．ただし，“リターンキーの力の限界”はこの場合にも存在していて，あまり調子にのって何でもかんでもマルチステートメントにしてしまっても，行の先頭から71文字までしか受け入れられませんから注意してください．

さて，ここで次のような計算式を考えてみましょう．

```
? 123/456*789 ……③
 212.822368
```

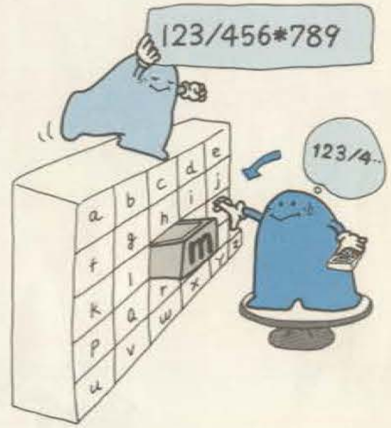
この答を変数 m に記憶させようと思ったら,

```
m=212.822368 ……④
```

を実行すればよいわけですね. この場合, まず答を計算し, 次にそれを変数に代入する, という 2 段階の作業を要したわけです. しかし, ただ変数 m に答を代入させたいだけならば, これは一回ですませることもできます. つまり,

```
m=123/456*789 ……⑤
```

というように, 代入命令の右辺に直接式を書いてしまえば, まずその式を計算して出てきた答を自動的に変数に代入してくれるのです.



変数に代入されている数を使いたいときには, その名前を呼べば答が返ってきます. 次の例を実行してみてください.

```
a=16
Ok
? a*2,4/a,SQR(a)      ……⑥
 32            .25
 4
Ok
```

このように, “a” と書かれている場所には全部, 16 という数があてはめられて計算がされます. a に代入する数を変えれば, こんどはその数があてはめられることになります.

```
a=100
Ok
? a*2,4/a,SQR(a)
 200           .04
 10
Ok
```
⑦

いままで出てきたことを組み合わせれば，次のようなこともできます．

```
a=6000:b=a+1 ……⑧
```

これを実行すると，変数 b の内容はいくつになるかわかりますか．そう，6001 が正解です．それでは次にいくつか例をあげますから，それぞれ最後に表示される数はいくつになるか，当ててみてください．




```
a=7:b=a+1:c=b*2:? c ……⑨
a=5:b=a+2:a=b+1:? a ……⑩
x=3:x=x*5:? x       ……⑪
m=1:m=m+1:? m       ……⑫
```

実際に PC に実行させてみて，きみの答が合っていたかどうか確かめてください．みんな当たりましたか？　はじめは当たらなくても，この章を何度も読んでいるうちに，きっとわかるようになりますよ．

第4章　文章を扱う



いままで見てきたように，BASICは数を計算するということについては，かなりの能力を持っています．もっとも，これはBASICに限らず，コンピュータなら，まあ数が計算できるのは当たりまえ．BASICにはその他に，文字や文章を扱うのがとても簡単だ，というすぐれた特徴があるのです．

この章では，その文字や文章を扱うためのBASICの機能について，簡単に使い方を説明していきましょう．けれども文字の取り扱いに関しての本当におもしろい部分は，残念ながらまだこの章では出てきません．それはまた後のお楽しみ．この本の第2部が始まるまで，しばしのお待ちを願いましょう．



4−1　文章を表示するには

```
? 2+3 ……①
```

この①を実行すると，どんな答が出てきますか．第3章を終えたきみには，もうこの答は，すぐにわかりますね．2たす3の答，5が表示されます．

```
 5 ……①の結果
Ok
```

さてそれでは，次の②のように，計算をダブルクォーテーションマーク（"）で囲んで実行すると，どんな結果になりますか（ダブルクォーテーションは，[" 2]のキーをシフトすると，キーインできます）．



```
? "2+3" ……②
```

この②を実行すると，2たす3の答ではなくて，“2+3”という文字列そのものが表示されます（いくつかの文字が並んでいるものを“文字列”といいます）．BASICでは，このようにダブルクォーテーションで，はさまれた文字列を，？に続けて打ちこむと，その文字列がそのまま画面に表示されます．

```
? "I am PC-6001"  ……③
? "1+1=3"         ……④
? "100000000000"  ……⑤
? "432e10"        ……⑥
```

文字列の中味は，③のように文章でもよいし，④のようなウソを書いても，まったくかまいません．また，⑤や⑥はダブルクォーテーションを取り除いて，数として表示させると，それぞれ浮動小数点表示に直されてしまいますが，このように文字列とした場合には，あくまでもそのまま，きみが打ちこんだ文字のまま，画面に表示されることになります．

さらに，数を表示するときには，その左に1文字ぶん符号のための空白がつく，ということを覚えているでしょう．しかし文字列の場合，間にそのような空白がはさまれることはありません．

```
? 10;20;30;40 ……⑦
 10 20 30 40
Ok

? "10";"20";"30";"40" ……⑧
10203040
Ok
```

つまり⑧のように，セミコロン（；）で区切った文字列は，互いにすき間なくぴったりとくっついて表示されるわけです．また，数を表示するときのように，表示したい文字が画面の右端までに入りきらないからといって，勝手に次の行の先頭まで進んでしまうこともありません．

```
? 8^4;8^5;8^6;8^7;8^8          ……⑨
 4096 32768 262144 2097152
 16777216
Ok
```

```
? " 4096";" 32768";" 262144";" 2 ……⑩
097152";" 16777216"
 4096 32768 262144 2097152 16777
216
Ok
```

要するに，文字列はきみが表示したいと思った形を忠実に守ってくれるわけです．数を表示するときのように，勝手に形を変えたり空白をあけたりしないのです．

さてそれでは，次の命令を実行すると，どうなりますか．

```
M=6001 ……⑪
? "M"  ……⑫
```

だまされないように，この場合でも，出てくるのは6001ではなくて，Mの文字そのものですね．変数のMと，文字の“M”は，この際まったく関係がないのです．

文字列と数は，いっしょに表示することもできます．次にある⑬や⑭のように，ただ数を表示するのではなく，言葉や文字をそえて説明すると，より楽しく，そしてわかりやすくなりますね．

```
? "3861*259=";3861*259 ……⑬
? "へんすう M は";M;" です"  ……⑭
```

このように，文字列と数が交互に出てくる場合は，間にあるセミコロンを省略することもできます．もっとも，そうするとどこまでが文字列で，どこまでが数であったか，とてもわかりにくくなりますから，あまりおすすめはしません．

```
? "1+2"8-5"4"2*6"7" ……⑮
```

⑮のような書き方では，なにがなんだかわからないでしょう！

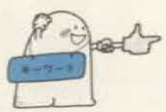
4－2　文字列を記憶する

第3章では，数を記憶するための"変数"というものについてお話ししました．同じように文字を覚えておくことができたらいいと思いませんか．キーボードを捜しながら，1文字，1文字打ちこんだ文字列が，クリアキーを押したとたんに消えてしまうなんて，悲しいでしょう？

それができるのです．コンピュータ用の言語は数ありますが，BASICはその中でも，文字列の記憶力に関しては，他のものにひけをとりません．それではさっそく実行してみましょう．

```
M="How are you?" ……①
```

ん？　おなじみの"ピッ！"という音が聞こえて，画面には次のようなメッセージが……，これはいけません．何か間違えたようですね．



```
?TM Error
```

そうです．変数Mです．変数Mは，数を記憶するために日夜働いているのですが，文字列まで覚えさせるのは荷が重すぎたのです．



文字列を記憶させるためには，文字列専用の記憶場所が必要です．それが"文字変数"です．文字変数は，普通の変数と同じような名前の後ろに"$(ドル)"をつけて表します．

```
M$="How are you?" ……②
```

これで，文字変数M$にHow are youという文字列が記憶されました．表示させて確かめてみましょう．

```
? M$ ……③
```

文字変数の名前は，最後に“$”がつく以外は普通の変数の名前のつけ方と同じです．つまり，アルファベット1文字の後に“$”がついたもの（A$～Z$），最初の1文字がアルファベットで，その後にいくつかの数字またはアルファベットが続き，最後に“$”がついたもの(A1$, AB$, ABC$など）が使えるわけです．どんなに長い名前でも，最後に“$”があれば文字変数だと判断しますが，やはり普通の変数の場合同様に，最初の2文字が同じ文字変数は，同じものだとされてしまいます．

```
MEMO$="QWE"     ……④
MENU$="123"     ……⑤
? MEMO$,MENU$
```

④，⑤の2つの文字変数はどちらもME$という文字変数と同じことになり，これらに別々の文字列を代入することはできないのです．

また，普通の変数の名前を書くときには，大文字でも小文字でもかまわなかったように，文字変数の名前も，大文字，小文字，どちらを使っても結果は同じです．ただし注意してください．大文字でも小文字でもかまわないのは文字変数の名前であって，文字変数の中味については，大文字は大文字，小文字は小文字となかなかきびしい区別があるのです．

```
A$="A"          ……⑥
B$="b"          ……⑦
? A$;a$;B$;b$
```



たとえば⑥，⑦を実行すると，A$の中味は大文字の“A”，B$の中味は小文字の“b”になりますが，これらの文字変数の名前を大文字にしようが，小文字で書こうが，その中味には影響していませんね．

次に，何も代入されていない文字変数について考えてみます．普通の変数では，何か代入されるまではすべてその内容は0でした．文字変数の場合はどうなっているでしょうか．まだいままでに使われていない文字変数，たとえばN$を見てみましょう．

```
? N$ ……⑧
```

何も表示されないようですね．もう少し，よくわかるように，両側に文字をつけてみます．

```
? "a";N$;"a" ……⑨
```

どうですか？　2つの"a"の間には何も文字はありませんね．つまりN$には何も入っていないということです．けれども，何もない数のことを"0"というように，この何もない文字列にも名前があって，"ヌルストリング"と呼ばれます．ダブルクォーテーションをすき間を開けずに2つ並べると，このヌルストリングを表すことができます．たとえば次のように，ヌルストリングを文字変数に代入する，ということも可能なわけです．




```
A$="" ……⑩
? A$
```

4－3　文字列の計算

さて，数に対しては変数があるように，文字列には文字変数が用意されている，ということはわかりました．それなら，数が計算できるように，文字列を計算するといったことはできないのでしょうか．答を先にいってしまうと，文字列や文字変数の間では，たし算だけが行えます．次の例を順に実行してみてください．




```
A$="これ"+"は"    ……①
? A$
B$=A$+A$          ……②
? B$
C$=B$+"おやまあ"  ……③
D$="おやまあ"+B$  ……④
? C$,D$
```

①のように文字列と文字列，②のように文字変数と文字変数，③，④のように文字列と文字変数と，いずれの組み合わせのたし算でも可能です．また，数のたし算の場合，1+2でも2+1でも答は同じ3になりますが，文字のたし算では③と④の例のように並び方によって結果は変わってきます．

よく間違えるのは，数字だけの文字列と本物の数字の区別です．いくら中味が数字だけだからと

いっても，ダブルクォーテーションにはさまれていれば，それは立派な文字列です．数を相手にして計算を行うことはできません．たとえば次にあげた例では，どれが間違いなのかわかりますか．そして，間違いではないとしたら，どんな結果が出てくるでしょうか．それぞれ試してみてください．

```
? "12"+"3"  ……⑤
? "12"+ 3   ……⑥
?  12 + 3   ……⑦
? "12 + 3"  ……⑧
```

ここで，ひとつの文字変数には，どのくらいの大きさの文字列まで入れられるのか，調べてみましょう．まずとりあえずは，とにかく長い文字列を文字変数に代入してみることから始めます．




```
A$="AAAAAAAAAAAAAAAAAAAAAAAAAA"  ……⑮
? A$
```

うん，どうやら26個の文字列は入ったようですね．他の文字変数でも同じでしょうか．やってみましょう．

```
B$="BBBBBBBBBBBBBBBBBBBBBBBBBB"  ……⑯
```

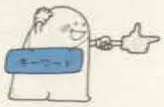
あれれ，こんどはエラーになりました．エラーメッセージは次のとおり．




```
?OS Error
```

実をいえば，PC では文字を記憶するための場所が，50文字ぶんしか用意してありません．それ以上の文字数を使おうとすると，"?OS Error"が現れてしまいます．この意味は，"文字列用の場所が，もうありません"ということになります．それにしても50文字しか使えないというのはあまりにも少なすぎる，と，きみは思うかもしれません．でも安心してください．そんなときには，きみの好きなだけ，文字列用の場所を用意する方法があります．たとえば1000文字分の領域を確保しておきたいと思うなら，次のように命令すればよいのです．

```
CLEAR 1000 ……⑰
```



このように，“CLEAR”の後に自分の使いたい文字数を書いて実行すると，PCはそれだけの場所を作り出します．“CLEAR”だけで後に数を書かなかった場合には，最後に指定したときの値を保ち，それでCLEARを実行します（こういうように，何かを省略したときに自動的に決まる値を，“ディフォルト”といいます．この言葉はこれからもたびたび出てきますから，覚えておいてください）．

気をつけなければいけないのは，CLEAR命令を実行すると，同時にそれまで記憶していた変数の内容，文字変数の内容がすべて消えてしまうことです．

```
A$="A"
X=100
CLEAR       ⑱
? A$;X
```

この⑱を上から順に実行していくと，最後ではA$はヌルストリング，Xは0となってしまっているのがわかるでしょう．

いま文字用の領域がどのくらいあるのか，ということを知ることもできます．それには次のように“FRE関数”を使います．



```
? FRE("") ……⑲
```

このとき返ってくる答が，残っている文字領域の大きさを示しています．文字列をたくさん使いたいという場合には，まずこのFRE関数で領域の大きさを確かめてから，CLEAR命令を使えばよいでしょう．なお，FRE関数のカッコの中には，ヌルストリングでなくても，文字列または文字変数であれば，実際には何を入れてもかまいません．FRE(a$) とか，FRE(“a”) などでも，同じ結果を得ることができます．

さて，今度は“CLEAR 1000”で充分な大きさの文字領域が用意されましたので，ひとつの文字変数にいくつの文字が入るのか調べられるでしょう．

```
A$="1234567890"
B$=""
B$=B$+A$  ) くりかえす     }⑳
? B$
```

⑳の4行を実行し，下の2行を何度もくりかえしてください．B$はどんどん長くなっていきますが，26回めになると，

```
?LS Error
```



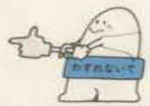
が出ます．これは“文字列が長すぎる”という意味のエラーメッセージです．どうやら文字列の長さは，250までは許されても260文字は代入できないようですね．さらにここで，

```
B$=B$+"12345"
```

を行ってみると，これはオーケー，けれどもこれ以上文字を加えることはできません．結局，ひとつの文字列に代入できる文字の数は255までということがわかりました．

4−4　ファンクションキー

キーボードのキーは普通，1個押せば1個の文字が出てくると決まっています．しかしキーボードの上に並んでいる F 1 ～ F 5 のキーだけは少し話が違います．これらのキーを押すと，1度にいくつものキーを押したときと同じ効果があるのです．この5つのキーのことを“ファンクションキー”といいます．



5個のファンクションキーはそれぞれ，シフトするかしないかによって別々の効果があり，合計10個の機能を持たせることができます．画面のいちばん下に表示されている文字は，実はこのファンクションキーを押したときにどんな効果があるかを示していたのです．たとえばここに表示されている文字のうち，右から2番目には“LIST”と書いてありますね．そこで右から2番目のファンクションキー F 4 を押すと，画面にも“LIST”と表示されるわけです．シフトキーを押すといちばん下にある文字は変化して，それぞれのキーをシフトしたときにキーインできる文字が現れてきま

す．

F 1 ～ F 5 をファンクションキーの1番から5番という番号で表し，それをシフトしたものをそれぞれファンクションキーの6番から10番で表します．たとえば F 6 と書けば，これは F 1 をシフトしながら押すことだと思ってください．

各ファンクションキーには次のようなキーを順に押したのと同じ効果があります．

```
 1 COLOR
 2 CLOAD"
 3 GOTO
 4 LIST
 5 RUN RETURN
 6 SCREEN
 7 CSAVE"
 8 PRINT
 9 PLAY
10 CONT RETURN
```

画面の下の表示では先頭から5文字までしか現れませんが，実際には2番や6番や7番のように6文字入っているキーがありますから注意してください．

いま10個のファンクションキーに入っている文字はそれぞれBASICでよく使われる命令なのですが，まだ意味もわからないし，こんなファンクションキーなんてあってもしかたないと思うかも知れません．しかし，ファンクションキーには，きみの好きな文字を覚えさせて使うこともできるのです．たとえば F 1 に“？A；B”と覚えさせてみましょう．それには次の“KEY”命令を使います．




```
KEY 1,"?A;B"
```

このように実行すると，画面の F 1 用の表示場所に“？A；B”という文字が表示され，これから F 1 を押せば，ここで指定した4文字が現れるようになります．変数AとBの内容を知ろうと思ったら， F 1 ， RETURN の2つのキーを押すだけでいいというわけです． さらに RETURN までここに覚えさせることができれば， F 1 1個押すだけで変数AとBの値を表示

させるようになります．しかしリターンキーは何度もいっているように，押してPCに命令を伝えるためのキーなので単にリターンキーを押しただけでは上のKEY命令が実行されてしまうだけです． リターンキーを覚えさせるためには，次のようにします．こんどは F 2 を使ってみましょう．

```
KEY 2,"?A;B"+CHR$(13)
```



CHR$(13)というのがリターンキーを表わしています．このKEY命令を実行したとき，画面の下の F 2 の表示は F 1 と変わらないように見えますが，押してみれば F 2 にはリターンキーの機能まで含まれたのがわかるでしょう．なお，このCHR$ (13) の他に，同じような方法で次のキーを表すことができます．

CHR$ (5)　CTRL + E
CHR$ (8)　デリートキー DEL
CHR$ (12)　クリアキー
CHR$ (18)　インサートキー INS
CHR$ (21)　CTRL + U
CHR$ (34)　ダブルクォーテーション（ " ）

ひとつのファンクションキーに覚えさせることができるのは，8文字までです．そのとき上にあるようなCHR$ (X) は，それぞれ1文字と数えてください．



第5章　プログラムの初歩



今までに説明してきたPCの使い方は，うまく頭にはいりましたか．きみも聞いたことがあるかもしれませんが，世界で最初のコンピュータというのはアメリカ合衆国で誕生し，その後もアメリカを中心に発達してきました．長い年月の間にはさまざまな種類のコンピュータが登場しましたが，今きみの目の前にあるPC-6001も，それら多くのコンピュータの中から生まれてきた子孫のひとつです．

そういうわけでコンピュータを説明しようとすると，どうしても英語やカタカナが多くなってしまうのですが，“ボクは英語は苦手なんだ”などと敬遠しないでください．トランプの“キング”や“ジョーカー”，“ハート”，“スペード”などの言葉だって英語なんですから，トランプってのは言葉が難しいからいやだなあ，なんて思わないでしょう？

さて，PCの中には，数を記憶するための変数や，文字列を記憶するための文字変数がそれぞれ何百個も用意されていて，いくつもの数や文章を覚えていられるということを，第3章や第4章では見てきました．けれどもPC-6001にはもうひとつ大切な記憶の能力が備わっています．それは仕事の手順を覚える，という能力です．どんなに長くて複雑な手順の仕事でも，いったん教えてやればPCは何度でも正確にそれをやってのけるのです．



このとき，PCに教える“仕事の手順”のことを“プログラム”といい，プログラムをPCに記憶させる作業を“プログラミング”といいます．これからきみがPCと付き合っていくときに，一番楽しくて同時に一番苦しいのがプログラミングの作業でしょう．そのプログラミングの世界への第1歩を，いよいよこれから踏み出します．心の準備はいいですか．

5－1　ひとつずつ実行，まとめて実行

いままでは，PC に命令を与えるとすぐにその場で実行しました．“1＋1を計算しなさい”—“ハイッ！”，“その答を覚えなさい”—“ハイッ！”という具合．このようにすぐにその場で命令が実行されるのは，“ダイレクトモード”（日本語でいえば“直接形式”）で命令を与えていたからです．なんだかダイレクトモードなんていう言葉を使うと難しく聞こえますが，要するにいままでやってきたことを，そう呼ぶのだと思ってください．



“ダイレクトモード”に対して，“プログラムモード”という方式があります．プログラムモードで命令を与えると，それはすぐには実行されず，仕事の手順の一部分として PC の中にしまいこまれます．そして最後にその覚えた仕事をいっぺんに実行しようというのです．

命令をプログラムモードで与えるのは簡単です．命令文の前にそれを実行させたい順番を書いておけば，それでよいのです．たとえば次のとおりに打ちこんでみてください．

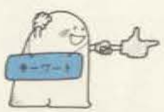



```
1 ? "こんにちは" RETURN
```

リターンキーを押してもなんにも起こりませんね．いままでだったら，たとえエラーになっても少なくとも最後に“Ok”と出てくることだけは決まっていましたが，それもありません．これは，PC がこの命令を“プログラム”として覚えたという印です．ここで続いて次のように打ちこんでみましょう．

```
2 ? "さようなら" RETURN
3 end           RETURN
```

2番の行，3番の行と，1行打ちこむごとにリターンキーを押すのを忘れないでください．ダイレクトモードで命令するときと同じように，プログラムモードでもリターンキーを押すことによって，本当に PC に命令が受け入れられるのです．



これで PC は，1，2，3の順に命令を実行する準備ができました（3番目の“end”というのは，そこで実行を終えなさいという命令です）．準備ができたら，さっそく実行させてみましょう．それ

には"RUN"という命令をダイレクトモードで行います.



```
RUN [RETURN]
こんにちは
さようなら
Ok
```



PCは1番目の命令と2番目の命令をまとめて実行しました(実際には3番目の命令も行ったのですが, これは目に見える結果を出さないのでわからなかったのです).

ダイレクトモードの命令は, 1度実行するともうそれで終わりの使い捨てでした. けれどもプログラムモードで打ちこまれた命令は, "プログラム"となってPCに記憶され, 何度でも繰り返して実行できるようになります. もう1度, RUN命令をダイレクトで実行してみてください. 再び同じ文が画面に表示されます. 3度目も4度目もまったく同じになることを試してください. もしこれをすべてダイレクトモードで行ったなら, キーインはとてもめんどうなことだったと思いませんか.

ここでひとつ便利な方法を紹介しておきます. ファンクションキーというのは, ひとつのキーでいくつものキーを押した場合と同じ効果を持っているものだと前にいいましたが, その中で一番右にある [F5] のキーを押すと, [R], [U], [N], [RETURN] の4つを順に押したときと同じ結果が得られます. つまり, [F5] キーを押すだけでRUN命令が実行できるわけです.



なお, RUN命令を実行することを, "プログラムを走らせる"または"RUNする"ということもありますから覚えておくとよいでしょう.

5－2 順序を示す行番号



プログラムを記憶させるためにそれぞれの行の先頭につける番号を"行番号"といいます. 行番号は, プログラミングを行うときに, とても重要な役割を持っています. これからその性質についてもっと詳しく説明していきましょう.

その前にまず, 次の命令を実行してください.

```
new [RETURN]
```



この“NEW”という命令は，いままでに記憶していたプログラムをすべて消してしまう働きを持っていて，新しく記憶させたいプログラムと古いプログラムが混ざってしまわないように準備をします．これからプログラミングを始めようというときには必ず最初にこの命令を実行する習慣をつけておくとよいでしょう．

さて，NEW命令を実行したら次のプログラムを打ちこんでください．

```
100 a=5 RETURN
101 ? "a=";a RETURN
102 end RETURN
```

それぞれの行でどんなことをやっているのかはわかりますね．まず変数aに5を代入し，それを画面に表示するというだけのものです．しかしこの行番号はどうも良くわからない．いったい1や2や3などの行番号はどこへいってしまったのでしょう．これでは1番目になにをやるのか，2番目にはどうするのか，100番めの行にある“a＝5”という命令が出てくるまでPCは困ってしまうのではないでしょうか．

けれど心配はいりません．PCは実際にある番号だけを見ていき，ない行番号は無視します．RUN命令で上のプログラムを実行してみましょう（F5キーを押すと簡単にRUNできるということはもういいましたね）．

```
RUN RETURN
a= 5
Ok
```

このとおり無事にプログラムは終わりました．ない番号を無視する例をもうひとつ見てみましょう（キーインする前にNEWを実行することを忘れないでください）．




```
new RETURN

144 ? "ごめん" RETURN
529 ? "      ください" RETURN
1156 end RETURN
```

こんなバラバラな番号をつけても，プログラムはちゃんと実行されます．

ただしひとつだけ規則があります．どのような番号の文をどんな順にキーインしても，実行のときは小さな行番号を持つ行から大きな行番号の行へ順にプログラムは進んでいきます．そのことを確かめるために，次のプログラムをこのとおりの順で打ちこんで実行してください．

```
new RETURN
20 ? "#20" RETURN
10 ? "#10" RETURN
30 end RETURN
```

キーインした順番とは逆に，"#10"，"#20"の順で表示されたでしょう．つまり最初に行番号10，次に行番号20の行が実行されたわけです．さてここで，NEWを実行しないで次の1行を打ちこんでみてください．

```
15 ? "#15" RETURN
```

これでRUNするとどうなりますか．こんどは"#10"，"#15"，"#20"というように表示されますね．つまりこの行番号15の行もいままであったプログラムに加わったのです．このようにNEWを実行しないでプログラムをキーインすると，それはどんどん元のプログラムの中に追加されていきます．

もしもすでにある行番号と同じものをキーインしたら，どうなるでしょうか．やってみましょう．

```
10 ? "#ten" RETURN
```

RUNした結果は，"#ten"，"#15"，"#20"となります．行番号10の内容は，あとから入れたものに変わってしまいました．

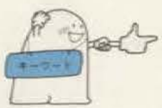
NEW命令は，プログラム全体を消すものですが，ある1行だけを消すこともできます．目的の行番号だけをキーインし，リターンキーを押せばその行はプログラム中から消えてしまうのです．たとえば，次のとおりに打ちこめば，いま記憶しているプログラムの行番号20の行は消えてなくなります．

```
20 RETURN
```

RUN してみると，"#ten"，"#15"とだけ表示されて，確かに 20 番の行は実行されていないことがわかります．

5－3 プログラムを見なおす

プログラムを追加したり消したりしていると，だんだん何がどうなっているのかわからなくなってきます．変数や文字変数ならば，いまどんなものが記憶されているのか，それを画面に表示させればすぐわかりますね．同じようにプログラムを画面に表示して，いまどんな状態になっているのか目で確かめられるといいと思いませんか．そのためにあるのが"LIST"です．




```
list ……①
```

この①を実行すると，5-2 で入れたプログラムが現れてきます．このように，PC の中にあるプログラムを実際の文字にして表わしたものを"プログラムリスト"または"リスト"といいます．




```
10 PRINT "#ten"
15 PRINT "#15"
30 END
Ok
```

ところでこの"リスト"を見ると，キーボードから打ちこんだ文字と変わってしまった部分があるのに気がつきます．まず目に入るのは"?"の文字がすべて"PRINT"という文字列にとってかわられていることでしょう．実をいえば，いままで数や文字列を表示するときに"?"を使ってきたのは略式の方法です．"PRINT"というのがプリント命令の正式な書き方なのです．



ダイレクトモードでちょこっと電卓のかわりに計算しようというときには"?"を使ったほうが手軽で便利ですが，プログラムを見るときには読んで意味がわかりやすい"PRINT"のほうがよいのです．そこでリスト中では"?"は自動的に"PRINT"に変えられるというわけです．

もちろん，ダイレクトモードで"?"ではなく"PRINT"を使ってもかまいません．たとえば

次の②と③はまったく同じことです.

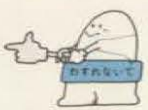
```
? 1+2        ……②
PRINT 1+2 ……③
```

もうひとつ，リストになったら変わってしまったものがあります．30行の“END”は，打ちこんだときは小文字の“end”だったはずですね．このように，リストの中ではアルファベットの小文字は大文字に変化します．ただしその場合，10行にあるように，文字列の中の小文字は変化しません．その他に，後で出てくる“REM文”と“DATA文”に含まれる小文字も変化しません．この3つの場合以外は小文字はすべて大文字になります．キーボードから文字を打ちこむときには，この3つ(文字列，REM文，DATA文)の場合だけ大文字と小文字の区別に注意すれば，他の場所ではどちらをつかっても結果は同じです．

LIST命令は，プログラムの一部分だけをリストすることもできます．次の例を実行してみてください．

```
LIST 15       ……④
LIST 10-15 ……⑤
```

④のように“LIST”の後に行番号をひとつ書いておくと，その行だけをリストします．また⑤のように2つの行番号をマイナス符号“－”で結んで書いておくと，その間にあるプログラムがリストされます．




```
LIST -15 ……⑥
LIST 15- ……⑦
```

2つの行番号のうち，⑥のように前にあるものを省略すると，プログラム先頭からリストされ，⑦のように後ろにあるものを省略すると，プログラムの最後までリストされることになります．

“LIST”の文字はファンクションキーの4番 F4 に入っていますから，これを使うと楽でしょう．ただし F5 の“RUN”とは違って，最後にリターンキーを押さないといけません．

5－4　プログラムの直し方

ここでひとつプログラムを打ちこんでもらいましょう．その前に古いプログラムは忘れずに消しておきます．NEW命令を実行すればよいのでしたね．

```
new
```

LIST命令を実行してみてください．プログラムはもう何も入っていないので，画面にはただ"Ok"と表示されるだけのはずです．このことを確かめたら，次のプログラムをキーインしてください．

```
100 a=7:b=25
150 ? "a+b =";a+b
200 end
```

このプログラムを実行すると，次のように表示されます．

```
a+b = 32
```

これでは，変数aとbはいくつなのかわかりませんから，下の2行を追加して，実行してみましょう．

```
130 ? "a =";a
140 ? "b =";b
```

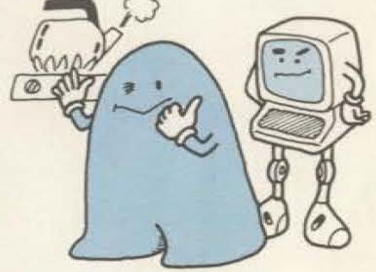
さて，いまリストをとるとどうなっているでしょうか．さっそくLIST命令を実行してみてください．

```
LIST

100 A=7:B=25
130 PRINT "a =";A
140 PRINT "b =";B
150 PRINT "a+b =";A+B
200 END
```

後から追加した130と140の2行は，ちゃんと100行と150行の間に入っていますね．どんな順に行を打ちこんでいっても，リストでは行番号の小さいほうから順に，つまりプログラムが実行される順に並べ換えてしまうのです．

さてそれでは，150行でたし算をやらせているところを，かけ算と取り換えたいときにはどうすればよいかわかりますか．確か，ある行を書き換えたいときには，新しい命令にその行と同じ番号をつけて打ちこむだけでよかったはずです．つまり，次のように打ちこめばよいわけですね．



```
150 ? "a*b =";a*b
```

もちろんこれで正しいのですが，正直にこのとおりすべてキーインする必要はありません．PCは，リターンキーを押したとき画面にある文字を命令として読みこむ，と前にいいましたね．そして，いまLIST命令を実行したため，画面にはすでに，

```
150 PRINT "a+b =";A+B
```

という文字が表示されているのです．これを利用しましょう．↑キーでカーソルを上に動かして，この150行に合わせ，さらに→キーでカーソルを右に動かし，“+”に合わせます．そこで“*”をキーインすれば，“+”は“*”に書き換わりますね．同じようにもうひとつの“+”も“*”に書き換えます．そこでリターンキーを押せば，これが新しく150行として記憶されることになります．リストをとって確かめてください．

このとき，LIST命令を実行するためには，再びカーソルを画面の下の何も書いてない部分に持っていくか，クリアキーで画面をクリアしておかないと，古いリストと新しいリストが重なってしまいますから注意してください．

さて，こんどは150行を次のように書き換えてリターンキーを押してみてください．

```
160 PRINT "a/b =";A/B
```

リストをとると，この160行がプログラムに入ったことがわかります．

```
100 A=7:B=25
130 PRINT "a =";A
140 PRINT "b =";B
150 PRINT "a*b =";A*B
160 PRINT "a/b =";A/B
200 END
```

このように，似たような内容の行を打ちこむときは，前からある行に少し手を加えるだけですみます．

インサートキーを使えば，いまある文字の間に別の文字を割りこませることができます．またデリートキーを使えば，いらない文字を消すことができます．この2つのキーの使い方はまだ覚えていますか．たとえば，次のような行を打ちこむことを考えてみましょう．

```
170 PRINT "a*a/b =";a*A/B
```

これは，160行に4つの文字を割りこませればよいわけです．まずカーソルを160行に合わせ，行番号を書き換えます．次にカーソルを“a”の文字に合わせ，インサートキーを1度たたきます．そして“a＊”とキーインすれば，その2文字が入りましたね．そうしたらまた[→]キーでカーソルを動かし，“A”に合わせ，同じようにインサートキーを押してから“a”と“＊”を順にキーインします．これでリターンキーを押せば，170行のできあがりということになります．

デリートキーの使い方は簡単でしたね．たとえば，170行を次の180行のように変えてみましょう．

```
180 PRINT "a*a =";A*A
```

この場合まずカーソルを170行の“b”のひとつ右の位置に合わせ，デリートキーを2回押せば，“／b”が消えるということになります．そして行の最後の“／B”を同様に消してしまえば，それでオーケー．おっと，ここで気をぬかないで，リターンキーを押すのを忘れちゃいけませんね．せっかく文字を書き換えたりしても，リターンキーを押すまではPCに何も伝わってないということを，よく心に止めておきましょう．

5－5　2つの行をくっつける

マルチステートメントの方法を使うと，1行にいくつもの命令を並べることができるといいました．たとえば130行と140行をいっしょにして，

```
130 PRINT "a =";A:PRINT "b =";B
```

とすることもできるわけです．これをひとつ試してみましょう．要するに，いまある130行の後ろに“PRINT"b　=";B”と打ちこめばよいのですね．けれども，また140行と同じものを手で打ちこむのは無駄というもの．140行をそっくりそのまま利用してしまいましょう．

まず，画面上で130行と140行をくっつけてしまいます．130行の右端にカーソルを持ってゆき→キーで130行から140行にかけてなぞっていくと,この2つの行は画面上ではひとつになってしまうということを思いだしてください．

このようにして2つの行をつないだら、カーソルを下の図の位置に合わせます．

```
130 PRINT "a =";A
140█PRINT "b =";B
   ↑
   カーソル
```

そしてデリートキーを押すと、140行にあった文字はカーソルに引きずられて、130行へ移ってゆきます．次の位置になったら止めてください．

```
130 PRINT "a =";A█PRINT "b =";B
                 ↑
                 カーソル
```

ここでコロン“:”のキーを押せばすべての文字は望みのとおりになりました．最後にリターンキーを押せば、それでできあがり，リストをとってみてください．

```
100 A=7:B=25
130 PRINT "a =";A:PRINT "b =";B
140 PRINT "b =";B
150 PRINT "a*b =";A*B
```

プログラム続く



```
160 PRINT "a/b =";A/B
170 PRINT "a*a/b =";A*A/B
180 PRINT "a*a =";A*A
200 END
```

おっと140行はまだ残っていますね．これはこれで消してやらないといけません．

```
140 RETURN
```

を実行してください．これでマルチステートメントへの変換は完了しました．

マルチステートメントを作るときには，わざと2つの行をつなげるわけですが，知らないうちに2行がつながってしまうということもよくあるものです．特に文字を打ちこんでいて，カーソルが画面の右端をうろついているときは要注意．たとえば今の130行の変数Bを，変数Cに書き換えたいとしましょう．

```
130 PRINT "a =";A:PRINT "b =";c█
150 PRINT "a*b =";A*B
                                ↑
                              カーソル
```

ここで手をすべらせて，あと1文字でも打ちこもうものならもう大変．130行とその下の150行はつながってしまうのです．もっともまだこの段階では，行がつながっているのは画面の上だけの話ですから，すぐに画面をクリアしてリストをとりなおせばよいのですが，このままリターンを押したら最後です．130行のあとに150行がくっついたおかしなものがプログラムとなって記憶されてしまいます．これでリストをとると，まるで150行が2つ出てきたように見えます．

```
130 PRINT "a =";A:PRINT "b =";C
150 PRINT "a*b =";A*B
150 PRINT "a*b =";A*B
```

このとき下に見えるのが本物の150行で，上にあるのはニセ物です．このままプログラムを実行すると，エラーになってしまうので，ニセ物の行は消さないといけません．それには，コントロールEを使うのが便利です．まず消したい行の先頭にカーソルを合わせてください．

```
130 PRINT "a =";A:PRINT "b =";C
150 PRINT "a*b =";A*B
150 PRINT "a*b =";A*B
```

そして CTRL キーを押しながら E のキーを押すと，カーソル以後の文字がすっかり消えてしまいますから，そこでリターンキーを押せば130行は正しく直されます．リストをとって確かめてください．

ただし，ひとつやっかいな問題が起こることがあります．いまの場合は，本物の150行は残っていてくれたのですが，操作のミスから，この本物が消されてしまう場合があるのです．

```
LIST 130

130 PRINT "a =";A:PRINT "b =";C
150 PRINT "a*b =";A*B
Ok
```

2行にみえるけれど，実は1行

このときにさっきと同じようにしてニセ物の150行を消してしまったら，あとから本物の150行は再び手で打ちこまなくてはなりません．それはちょっとめんどうですね．こんなときにはテレビ画面を活用して，次のような方法を使うとよいでしょう．まず画面に130行だけのリストを2つ表示させます．

```
LIST 130

130 PRINT "a =";A:PRINT "b =";C
150 PRINT "a*b =";A*B
Ok
LIST 130

130 PRINT "a =";A:PRINT "b =";C
150 PRINT "a*b =";A*B
Ok
```



そして，まずそのリストのうちの片方で，150行を消してください．これは前の方法と同じようにコントロールEを使えばいいでしょう．次にもうひとつのリストへカーソルを動かし，130行の分だけを消してください．これはデリートキーかスペースキーで1文字ずつ消していくしかありません．最後に両方の行にカーソルを合わせて，それぞれリターンキーを押せば，うまく2つの行に分けることができます．

5－6　打ちこんだはずなのに



マルチステートメントを使うとプログラムの1行にたくさんの命令を入れることができますが，1行に71文字までしか許されないのはダイレクトモードの場合と変わりません．

特にプログラムモードの場合に気をつけなくてはならないのが，"?"を使ったときです．キーボードから打ちこむときは71文字以内でも，リストにするとそれ以上の長さになることがあるからです．たとえば次のプログラムをキーインしてみましょう．

```
10 ?"aaaaa":?"bbbbb":?"ccccc":?"
ddddd":?"eeeee":?"fffff"
```

これは71文字よりずっと短いので，もちろん全部受けつけられます．しかしリストを表示させてみると，"?"が"PRINT"に変わるため全体として71文字を超えてしまいます．

```
10 PRINT"aaaaa":PRINT"bbbbb":PRI
NT"ccccc":PRINT"ddddd":PRINT"eee
ee":PRINT"fffff"
```

このままプログラムを実行するならば，何の問題もないのですが，このリストされている行にカーソルを合わせてリターンキーを押してしまうことがあると，そこで行の72文字目以後は切り捨てられてしまうのです．そのリストを見ると次のようになっているはずです．

```
10 PRINT"aaaaa":PRINT"bbbbb":PRI
NT"ccccc":PRINT"ddddd":PRINT"eee
ee":PRI
```

71文字目までで切れている

同じようなことはもうひとつあります．たとえば次のようなプログラムを打ちこんでみてください．これは，ちょうど71文字なので，ぎりぎり受けつけられるはずですね．

```
10A$="78901234567890123456789012
345678901234567890123456789012 34
":A$=B$
```

リストをとっても，ちゃんと最後の文字まで入っているでしょう．しかしよく見るとこのリストは72文字になっています．これはリストをとるときには必ず，行番号の後ろに1文字分のスペースをあけるようになっているからです．この行にカーソルを合わせてリターンを押すと、最後の“$”の文字が欠け落ちてしまいます．

```
10 A$="7890123456789012345678901
2345678901234567890123456789012 3
4":A$=B
```

最初にプログラムを打ちこんだときは，うまく実行されたものが，リストを見ているうちにいつの間にかエラーを出すようになってしまったというときは，これら2つのことが起こっていないかどうか，よく調べてください．

さて，そうしたら今度は次のようなプログラムを打ちこんでみましょう．

```
65500 a=1
65510 b=2
65520 c=3
65530 d=4
```

こんな短いプログラムなら，何の問題もなく打ちこめると思うでしょう．ところがそうではありません．リストをとってみてください．

```
6553 0 D=4
65500 A=1
65510 B=2
65520 C=3
```

65530行は入らずに，6553行とされてしまいましたね．実はPCでは，行番号として使える数字は0から65529までなのです．それ以上の行番号を使おうとすると，このようにおかしな結果が出てきてしまいます．打ちこんでいるときにはエラーとも何とも表示されず，リストをとって初めてわかることなので，行番号の大きさにはいつも充分注意していてください．

5－7　間違いを見つけ出すには

ダイレクトモードで命令を実行させる場合は，何かまずいことがあれば，その場ですぐにエラーメッセージが出されます．しかしプログラムモードで打ちこんでいるときは，どんなに間違った命令でもそのまま記憶され，実行してみて初めてエラーとわかるのです．もしも単にエラーメッセージが出るだけでは，プログラムのどこに間違いがあるのか捜すのが大変でしょう．

そこでPCは，プログラムを実行していてエラーが起きたときには，エラーメッセージの後ろにそのエラーが起こった行の行番号を表示してくれます．たとえばもし100行に“A$＝1”のような間違いがあったとしたら，

```
?TM Error in 100
```

のようにエラーメッセージの後ろに“in 100”という行番号を示す文がついてくるわけです．

さてそこで，もしある行に間違いがあるとわかったらどうすればよいでしょうか．まずその行をリストに表示してみます．そしてメッセージの種類によって，それぞれどんな手を打てばよいか考えるのです．次にいくつか例をあげてみましょう．



? SN Error の場合

プログラムを打ち間違えています．よく見ればどこか文字が違っていたり足りなかったりする場所があるはずです．それを直してください．



? MO Error の場合

やはりプログラムを打ち間違えています．特に数字や文字列があるはずの場所に何も書いていな

いというときに出るエラーです．“+”や“＊”や“=”など，アルファベット以外の記号の後ろに注意してください．

? OV Error の場合



計算の結果が，1.7 E 38 より大きくなってしまいました．そんな計算をやらせてはいけません．もしその行の中に変数が出てきていたら，ダイレクトモードでその内容を表示させてみると原因となった場所がわかることがあります．

? TM Error の場合



文字と数がごっちゃになっています．文字列と数の間で計算していないか，文字変数に数を代入していないか，変数に文字列を代入していないか調べてください．

? FC Error の場合



許されている範囲外の大きさの数を使ってしまいました．許されている範囲といっても場合によって異なります．SQR (X) のカッコの中が負の数になっていないか，TAB (X) のカッコの中が 256 以上ではないか，などそれぞれ確かめてください．

?/0 Error の場合



0 でわり算をしてはいけません．その行の中でわり算が出てきていたら，そこに原因があります．とても珍しいことですが，TAN 関数の引数がちょうど $\pi/2$ になって，このエラーが起きることもあります．

? LS Error の場合

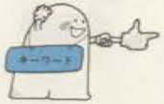


文字列の長さが 256 を超えました．文字列のたし算をしている場所に原因がありますから，そこに注意してください．



? OS Error の場合

文字列用の領域が足りなくなってしまいました．“CLEAR”命令を実行して，文字領域をもっと大きくとってください．



? OM Error の場合

プログラム用の領域が足りなくなってしまいました．CLEAR 命令であまり文字領域を大きくとりすぎると，こんどはこっちのエラーになることもあります．いまどのくらいのプログラム用領域が残っているのかは，“? FRE(0)” で知ることができますから参考にしてください．

プログラムの中にある間違いを “バグ” または日本語で “虫” といいます．そしてこの “虫” を取り除くことを “デバッグ” といいます．もしもプログラムに “バグ” がいたら，いまあげたようなエラーメッセージの意味を考えながら，確実にデバッグしていってください．

5－8　記憶にございません？

まず次のようなプログラムを打ちこんでください．

```
new
100 PRINT V*V
110 END
```

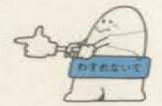
これを実行するとどうなるでしょうか．そんなことといわれても，変数 V の内容がいくつなのかわからなければ答えられないって？　うん，いいところに気がつきましたね．それなら何か数を決めて代入しておきましょう．とりあえず 6 を代入することにします．

```
V=6
```

これでさっきのプログラムを実行するとどうなるか答えられますね．36 になる？　自信ある？　それでは PC に答えてもらいましょう．

```
RUN
 0
Ok
```

残念でした．0になります．実はプログラムをRUNすると，それまでダイレクトモードでいっしょうけんめい覚えこませた記憶はすっかり消えて白紙の状態になってしまいます．数の変数はすべて0になり，文字変数はヌルストリングに変わるのです．



ただし，この逆ならばオーケー．つまりプログラム中で計算した答は，プログラムが終了しても変数の中に残っていて，ダイレクトモードで表示させることができます．次のプログラムを実行してみてください．

```
100 A=6:B=A*A:C=B*B
110 A$="ひっ":B$=A$+A$
120 END
```

そうしたら，ダイレクトモードで，変数の内容を見てみましょう．

```
? a$;b$;a;b;c
ひっひっひっ 6 36 1296
Ok
```

というように，ちゃんとまだ覚えています．

とにかく，プログラムをRUNすると，変数や文字変数の内容は消えてしまう．このことは忘れないでください．

5－9　プログラムをとっておくには



せっかく打ちこんだプログラムも，PCの電源を切れば消えてなくなってしまいます．また，電源を切らないとしても，新しいプログラムを入れるなら，古いプログラムは消してしまわなくてはなりません．ほんの数行のプログラムだったら，消えてもまた手で打ちこみなおすのに，そんなに苦労もいりませんが，とても長いプログラムで打ちこみに何時間もかかるようなものでは，消えてなくなるのはちょっと待ってもらいたいですね．

そういう場合のため，PCではテープにプログラムを保存できるようになっています．テープといっても，普通の音楽用カセットテープとカセットテープレコーダでかまいません．ただし，確実さを求めるならば，データレコーダ（PC-6081，PC-6082）を使ったほうがよいでしょう．

用意するもの

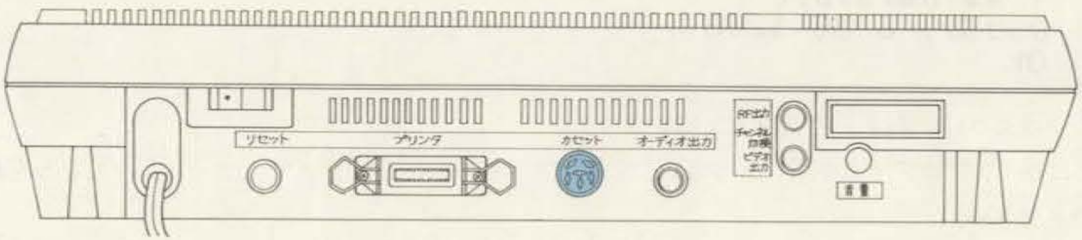
① テープレコーダ（以下テレコといいます）

② 録音テープ

③ テレコとPCをつなぐための接続コード …PC-8093
（片方DINプラグ，もう片方がミニプラグとなっているもの）

それでは，プログラムをテープに保存する方法を説明しますが，その前に何か2，3行でいいですからプログラムをPCに入れておいてください．

さてそうしたら，まず接続コードのDINプラグをPCの背面にあるソケットにさしこみます．次に赤いミニプラグをテレコの録音用ジャックにさしこみます．これで接続は終わりました．



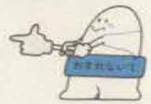
次にテープをテレコにセットします．テープの端に磁性体の塗られていない透明の"リーダー"と呼ばれる部分が長くついているものは，その分を手で巻き進めて茶色の磁性体の部分を出しておいてください．

最後に，キーボードより次のように打ちこみ，同時にテレコの録音ボタンを押してください．



CSAVE"TEST" RETURN

このようにプログラムを保存することを，プログラムを“セーブ”するといいます．セーブするプログラムには名前をつけて区別します．上の例では“TEST”というのがその名前です．こういう名前のことを“ファイルネーム”といいます．ファイルネームは6文字までしか受けつけられず，それ以上書いても無視されます．たとえば，



```
CSAVE"12345678"
```



としても，実際には123456という名前になってしまうわけです．

プログラムのセーブが終わると，PCは画面に

```
Ok
```

と表示します．まずたいていはこれで大丈夫なのですが，念のためいまセーブしたテープのすぐ後にもう一度同じプログラムをセーブしておくとよいでしょう．

テープに保存したプログラムは，またPCの中に読みこめなければ意味がありません．それを行うのが“CLOAD”命令です．



まず，CLOAD命令で確かにプログラムが読みこめるのかどうか知るために，いまあるプログラムはNEW命令で消しておいてください(実際にはCLOAD命令を行うと自動的にNEW命令が実行されるので，あらためてNEWを行う必要はありません)．

そうしたら接続コードの白いミニプラグを，テレコのイヤホーン・ジャックにさしこんでください．そしてさっきプログラムをセーブしておいたテープを巻き戻し，キーボードから

```
CLOAD
```

または，

```
CLOAD"TEST"
```

と打ちこんで実行します．テープからプログラムを読みとることは，プログラムを“ロード”するといいます．ファイルネームはつけてもつけなくてもかまいませんが，一本のテープにたくさんの



プログラムがセーブされているような場合，ファイルネームをつけて CLOAD 命令を実行するとその名前のプログラムを捜し出してロードしてくれます．ファイルネームがないときには，最初に見つかったプログラムをロードします．

テレコの再生ボタンを押してしばらくすると，うまくいった場合には画面に




```
CLOAD
Found:TEST
```

と表示され，画面の右上でアスタリスク（＊）が点滅を始めます．そしてしばらくたって

```
Ok
```

の表示が出れば，それでオーケイ．リストをとってみるとさっきセーブしたプログラムが PC の中にまた記憶されたことがわかるはずです．

しかし，もしなんにもロードできなかったり，ロードはしても，



?TR Error

が表示されたときには，テレコのボリュームを調節して何回も挑戦してみてください．そのときだいたいのめやすとして，きみの耳で音を聞いてみるという方法があります．白いプラグをテレコからいったん抜いてプログラムの部分を再生してみれば，プログラムが音となって聞こえてくるでしょう．最初“ピー”が1秒くらい続き，その後“ガジャガジャ”という音が入っています．この音は小さくてもだめですが，耳が痛いほどの大音量でもやはりいけません．きみにとって，やや大きいかな？　という程度の音にボリュームを合わせれば，きっとその近くに PC がうまくプログラムをロードすることのできるポイントがあるはずです．

1度うまくロードできたら，忘れないようにテレコのボリュームに印をつけておきましょう．以後はその印にボリュームを合わせてロードすればよいわけです．

ちゃんとロードもできたし，“Ok”も表示された，というときでも，よく見るとプログラムの中の何文字かが間違えられていることがあります．これはセーブをしたときに雑音が混ざり，そのまま

録音されてしまったのです．こういうことを避けるために，２回続けてセーブしたりするわけですが，正しくセーブされたかどうかPCに確かめさせる方法もあります．

プログラムをセーブし終わったら，すぐに巻き戻して次の命令を実行してください．

```
CLOAD?
```



ロードするときのCLOAD命令と似ていますが，この場合ロードは行われずに，PCの中に記憶されているプログラムとテープに録音されているプログラムが照らし合わされます．そしてもしどこか違っていれば，

```
Found:TEST
Bad
```

と表示され，すべて合っていれば

```
Found:TEST
Ok
```

と表示されます．この命令でもCLOADと同じようにファイルネームをつけることができます．やはりこのときも最初に見つけたプログラムを対象にして照らし合わせが行われます．目的のプログラムが見つかるまでに別のプログラムがあった場合には，そのプログラムのファイルネームを

```
Skip:あいうえお
```

のように表示します．これはCLOADの場合も同じです．

テレコとPCの接続コードには，赤と白の他にもうひとつ，黒いプラグがついています．これはリモートコントロール用の端子で，リモート用のジャックがついているテレコでは，このプラグをそこに指しこんでおくとPCが自動的にテレコのスイッチを切ったり入れたりできるようになります．



“リモート端子”をつながないでおくと，セーブやロードが終わっても，テープはどんどん回りつ

づけますが，つないでおくとそこでPCが止めてくれるのです．これを使うと一本のテープにたくさんのプログラムを効率よくまとめることができます．ただし，このリモート端子をつないでしまうと，テープの早送りや巻き戻しをすることができないので，そのときはプラグを抜いてから行ってください．

なお，データレコーダ(PC-6081，PC-6082)は，リモート端子をつないでもテープの早送りや巻き戻しができるので，リモート端子をつないでおくと便利です．



第6章　プログラムを楽しもう



プログラムを与えてやれば，PC はきみの思うとおりの仕事をしてくれます．けれども第5章までに出てきたのは，数や文字を表示するための“PRINT”命令と変数や文字変数に代入を行う“代入”命令，プログラムを終了させる“END”命令の3つだけで，これではあまりおもしろいことや，本当にコンピュータらしい便利な働きをさせることはできません．

以下この第6章では，BASIC が持っているいろいろな命令を順に説明していきます．これらの命令を使うことを覚えれば，おもしろいプログラム，便利なプログラムをたくさん作ることができるようになります．そうなれば，きみにもプログラミングの楽しさが，だんだんとわかってくるでしょう．

6－1　言ってくれなきゃわからない

まず次のプログラムを打ちこんでください（キーインする場合は，このリストどおりではなくて，小文字や“？”を使ってもかまわないんでしたね）．

```
100 A=20
110 S=A*A
120 PRINT A;"*";A;"=";S
130 END
```

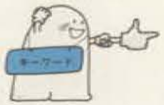
これは 20＊20 を計算するものです．ファンクションキーの F5 を押して，実行してみてください．

```
RUN
 20* 20= 400
Ok
```

プログラムの100行を書き換えれば，20以外の数でも計算できます．たとえば，100行を次のように変えてから実行してみましょう．

```
100 A=307

RUN
 307* 307= 94249
Ok
```



しかし，数を変えて計算するたびにリストを直すのでは大変ですね．こんなときに使うと便利なのが“INPUT命令”です．こんどは100行を次のように直してから実行してみてください．どうなるでしょうか．




```
100 INPUT A
```

このプログラムをRUNすると，画面には“？”の文字が現れ，その後ろでカーソルが点滅を始めます．

```
RUN
? █
```

ここで，“20”と打ちこんでリターンキーを押してください．すると画面には次のように答が出てくるはずです．

```
 20* 20= 400
Ok
```

どうやら打ちこんだ数がそのまま使われたようですね．何回もRUNしていろいろな数を打ちこみ試してみてください．

プログラムを実行している途中でINPUT命令に出会うと，PCはそこでいったん止まって，“いくつですか？”ときみに質問してきます．それを示すのが“？”の文字です．そしてきみが数をキー

インしてやれば，プログラムリストの中の“INPUT”の後ろに書いてある変数にそれが代入されるのです．“A=20” などの代入命令では，変数に入るのはいつでも決まった数でしたが，INPUT 命令を使うと，RUN することに違う数を選んで変数に代入することができるわけです．

INPUT 命令を実行するときのように，PC がカーソルを点滅させて待っている状態を“入力待ち”の状態といいます．入力待ちには 2 つの種類があって，ひとつはこの INPUT 命令での入力待ち，もうひとつは PC が何もしていないときの入力待ち，つまりプログラムやダイレクトモードの実行命令が入力されるのを待っている状態です．この 2 つは見た目にはまったく同じで，よく間違えることがあります．



たとえば，いまのプログラムを実行させて入力待ちの状態になったら，クリアキーを押して画面を消してみましょう．これでもまだ PC は INPUT 命令の入力待ちをしているのには変わりないのですが，入力をうながす“？”が消えてしまったので，ますますどっちの入力待ちなのかわかりません．もしここで勘違いをしてプログラムを打ちこもうとしたり，RUN 命令を実行しようとすると，PC は困ってしまって，

```
?Redo from start
? █
```



というメッセージを出します．これは，“INPUT 命令にもう 1 度答えてください”という意味で，INPUT 命令に対しておかしな答を入力したときに出されるものです．変数 A に代入できるのは数だけですから，この場合は数以外の答を入力すると，“? Redo from start”が出てしまうのです．



次に INPUT を使ったプログラムの例をもうひとつあげてみましたので，実行してみてください．これは 2 つの数でわり算を行い，答と余りを表示するものです．

```
100 INPUT A:INPUT B
110 C=INT(A/B):D=A-B*C
120 PRINT A;" わる";B;" は・・・"
130 PRINT C;"  あまり";D
140 END
```

実行してみてください．この場合には，2 回にわけて数値を入力しなければなりません．

このプログラムの100行のように，いくつかのINPUT命令が並んでいるときは，それをひとつにまとめることもできます．100行を次のように書き直して実行してください．

```
100 INPUT A,B
```



変数と変数を区切るのはカンマ（,）です．このようにしたときには，数を入力するときも2つまとめて行えます．これはやはり2つの数をカンマで区切ってキーインすればよいのです．

```
RUN
? 654,321
 654 わる 321 は・・・
 2  あまり 12
Ok
```

また，このときに1個だけしか入力しないと，“？？”のようにクエスチョンマークが2つ出てきて，次の数を入力するようにうながします．逆に数を2個よりも多く入力してしまった場合には，



```
?Extra ignored
```

というメッセージが表示されます．これは“多すぎた分は無視します”という意味です．

もし，“？？”が出ているときに間違った入力を行って“? Redo from start”のメッセージが出されると，再び1番目の変数への入力から始めることになるので気をつけてください．



普通，プログラムを実行していてエラーが起きると，そこでプログラムは終了してエラーメッセージが出されるのですが，INPUT命令のときは“? Redo from start”も，“? Extra ignored”もメッセージだけ出してプログラムは止まりません．これは人間がキーインを間違えたときにいちいち止まってしまい，また最初からやりなおすのではプログラムがなかなか先へ進まないからです．そうはいっても，やはりエラーが出るのはいやなものです．しかし入力のときにただ“？”という文字が出てくるだけでは，いったい何を求められているのかわからずに，適当でない入力をしてしまうこともあるでしょう．そこで，INPUT命令では，最初に好きな文字を表示しておいてから入

力を行わせるという方法が使えるのです．100行を次のとおりに直してから実行してください．

```
100 INPUT "A,B は いくつ";A,B
```



これを実行すると，次のように表示されます．

```
RUN
A,B は いくつ? █
```

つまり“INPUT”の後に文字列を書くと，まずそれが画面に表示されてから，INPUT命令が実行されるのです．このとき表示される文字列を“プロンプト文”といいます．プロンプト文用の文字列と変数とはセミコロン（;）で区切ります．また，ひとつのINPUT命令には，ひとつのプロンプト文しかつけられません．もしも変数のそれぞれに入力を行うごとにメッセージを出したいのだったら，INPUT命令は2つに分けなければなりません．たとえば次のようになるわけです．




```
100 INPUT "A は いくつ";A
105 INPUT "B は いくつ";B
```

さて，INPUT命令で数を変数に代入できるなら，文字列を文字変数に代入することもできるのではないか，と思うでしょう．そのとおり．できるのです．次のプログラムを実行してみてください．




```
100 INPUT "あなたは だ゛あれ";A$
110 PRINT A$;" くん、 よろしくね。"
120 END
```

INPUT命令で文字変数に文字列を代入するのは，普通の変数の場合とまったく同じやり方です．ただの変数のかわりに文字変数を使えばいいだけです．

さて，上のプログラムをRUNしたら，次のように入力するとどうなりますか．

```
あなたは だ゛あれ? super man
```

小文字で入力すれば，それは小文字のまま，文字，数でも，かな文字でも，キーインしたとおりに文字変数に代入されますから，試してみてください．

ところがここにひとつだけ例外があります．文字列の先頭にある空白と，最後にある空白は無視されてしまいます．プログラムをRUNして，次のように入力してください．

```
あなたは だ゛あれ?　　　　　　　　およよ
```

最初にいくら空白をつけても，それはまったく無駄ですね．もしも文字列の先頭や最後に空白をつけて入力したいのなら，それはダブルクォーテーションでくくってやる必要があります．次の例を見てください．




```
RUN
あなたは だ゛あれ? "　　 およよ　　 "
　　 およよ　　　 くん、 よろしくね。
```

2つのダブルクォーテーションにはさまれた部分がみんな，文字変数に代入されたのがわかるでしょう．ただしダブルクォーテーション自身は文字列に含まれません．

最後にもうひとつだけ，何も入力しないでリターンキーを押したらどうなるか，確かめておくことにします．いまのプログラムで入力待ちのとき，ただリターンキーを押せば，次のようになります．

```
RUN
あなたは だ゛あれ?
 くん、 よろしくね。
Ok
```

どうもただリターンキーを押しただけではA$にヌルストリングが入るようですね．ところが次の行を追加すると結果は違ってきます．




```
90 A$="ななしの ご゛んべ゛"

RUN
あなたは だ゛あれ?
ななしの ご゛んべ゛ くん、 よろしくね。
Ok
```

つまり，INPUT 命令の入力待ちのとき，何も入力しないでリターンキーを押すと，そのとき指定された文字変数には変化がなく，そのまま INPUT 命令を通りぬけてしまうのです．これは普通の変数でも同じこと．次のプログラムを実行してみてください．

```
100 A=6001:INPUT A
110 PRINT "PC-";A
120 END
```

何か数を入力すれば，ちゃんとその数が表示されますね．しかしリターンキーだけ押すと，最初に代入している 6001 が表示されるということになるわけです．

6－2　流れを変える，流れを止める

いままでのプログラムは，繰り返して何度もやろうと思ったら，いちいち RUN 命令を実行しなければなりませんでした．けれどもどうせ何度も行うプログラムだったら，繰り返しは PC 自身にやらせてしまいましょう．次のプログラムを見てください(これは 50 m 走るのにかかる秒数から，きみの時速を計算するプログラムです)．




```
100 INPUT "50m なん秒";T
110 H=50/T*3600/1000
120 V=INT(H*100+0.5)/100
130 PRINT "きみの はやさは 時そく";V;"km です。"
140 GOTO 100
```

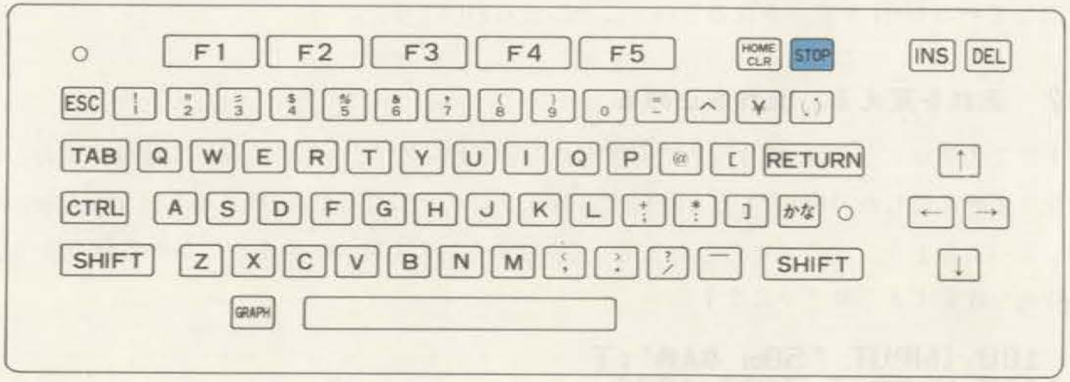


140 行に新しく出てきたてきたのが "GOTO" 命令です．この命令に行きあたるとプログラムの流れは "GOTO" の後ろに書いてある行番号のところへ移り，こんどはそこから別の道を進みはじめることになります．プログラムの "流れ" というのは実行の順序のことです．普通は行番号の小さいほうから大きい方へ流れますが，それだけでは思いどおりのプログラムを作ることができません．そこで，BASIC には実行の順序を思いどおりに変える，つまり流れを変える命令がいくつか用意されているのです．

さて，話を戻しましょう．さっきのプログラムの場合だったら，“GOTO 100”なので，次は100行に移るわけです．そしてそれがまた140行へ進んできては100行に戻るという繰り返しがいつまでも続きます．プログラムの流れがこのように輪を描いたようになることを，プログラムが“ループ”をつくる，といいます．特にこのプログラムのように永久に止まらないループは，“無限ループ”といいます．



しかしいくらなんでも，本当に永久に止まらないのではこまりますね．実はこの無限ループから抜け出るための方法はちゃんとあるのです．下図の位置にある STOP というキーがその秘密です．





このSTOPのキーを“ストップキー”といいます．ストップキーはなにも無限ループに限らずどんなプログラムを実行しているときでも，押せばすぐにそのプログラムを中断する働きを持っています．ストップキーを押してプログラムを中断することを，“ブレイクをかける”ともいいます．それではいま動いているプログラムに，さっそくブレイクをかけてみましょう．そうすると画面に




```
Break in 130
Ok
```

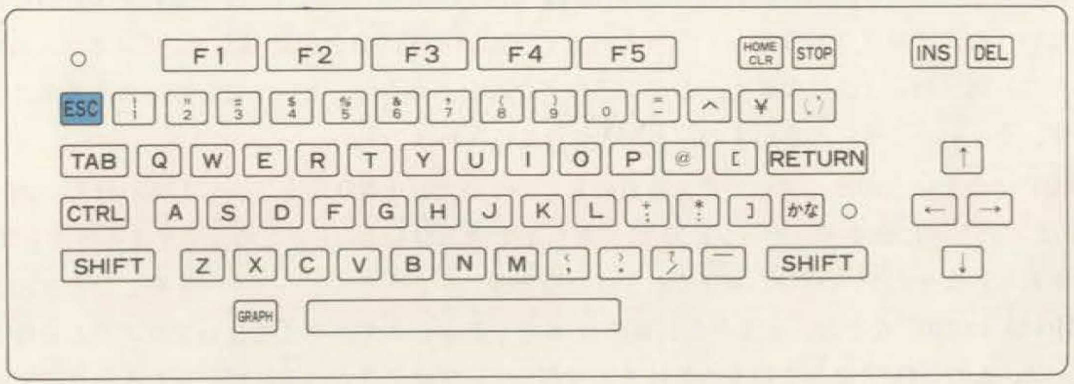
という表示が現れましたね．130というのはブレイクがかかったときの行番号ですから，きみは違ったかもしれませんが，心配しないでください．PCはプログラムの入力待ちに戻っていますね．

無限ループの例をもうひとつ見てみましょう．次のプログラムを打ちこんで，実行してください．

```
100 PRINT A
110 A=A+1
120 GOTO 100
```

RUN すると，0から順にひとつずつ数が表示されはじめ，いつまでたっても止まりません．さっきの例よりもずっと“無限ループ”という言葉の感じが現れているように思いませんか．

さて，このプログラムをストップキーで中断させることももちろんできるのですが，ここで別のキーを使ってみることにしましょう．下図の位置にある ESC キーを押してみてください．すると動いていた画面がピタッと止まるでしょう．このキーを“エスケープキー”といいます．エスケープキー以外のキーを押すと，画面はまた動きだします．このエスケープキーは，ストップキーのようにプログラムから完全に抜け出すのではなく，ただ一時停止を行わせるためのものです．特に画面が動いてよく目に止まらないようなとき，いったん止めてじっくり観察するのに便利でしょう．





ストップキーが外からプログラムを止めるのに対し，プログラムの中にあってプログラムを止める命令が2つあります．ひとつはすでにおなじみの“END”命令，もうひとつはストップキーと同じ名前の“STOP”命令です．



END 命令はいままで，最後に置いてプログラムを終了させるためにあると説明してきましたが，これはあまり意味のないことでした．プログラムの最後に END 命令がなくても，最後の行に達

すれば，実は自然にプログラムは終了するのです．たとえば次のようなプログラムでは，130行のEND命令はあってもなくても同じです．

```
100 PRINT "0"
110 PRINT "1"
120 PRINT "2"
130 END
```



しかしEND命令はプログラムの間に出てくるときには大きな意味を持ちます．次の例を見てください．

```
100 PRINT "0"
110 END
120 PRINT "1"
130 PRINT "2"
```

これをRUNしても，実際に実行されるのは110行のPRINT命令だけで，120行と130行では何もしないうちにプログラムは終わってしまいます．特にGOTO命令が登場して，プログラムの流れがあちこち跳び回るようになると，どこでプログラムを止めるか，ということはとても大事になるのです．そういうときに力を出すのがEND命令というわけです．



STOP命令というのは，プログラムを止めるという点ではEND命令と少しも違いはないのですが，止まったときに出すメッセージに特徴があります．END命令は，どこの行で止まろうとも“Ok”としか表示しませんが，STOP命令で止まった場合は，ちょうどストップキーを押したときのように“Break in 130”などと，止まった位置の行番号を表示します，そこで長いプログラムを作るときなどは，このSTOP命令をいくつかはさんでおいて，プログラムの流れが思ったとおり進んでいるのか調べるのに使ったりします．ともかく，止まったときの行番号を知りたいときにはSTOP命令，そうでないときはEND命令，と使いわければよいでしょう．

プログラムを止める3つの場合，つまりストップキーを押すか，END命令を実行したか，STOP命令を実行したか，このどの場合でも，中断させたプログラムは再開することができます．そのための命令が“CONT”命令です．プログラムが中断しているときにこの命令をダイレクトモードで



実行すると，止まっていた次のところからまたプログラムは始まります．たとえば次のプログラムを実行してみてください．

```
100 PRINT 1
110 END
120 PRINT 2
130 STOP
140 PRINT 3
150 GOTO 140
```

RUN すると，まず 100 行が実行され，110 行で止まります．そこで CONT を実行すると，120 行を実行して 130 行で止まります．再び CONT を行うと 150 行のループに入ります．後はストップキーを押すとブレイクがかかり，CONT を行うと再開する，の繰り返しということになります．

ファンクションキーの 10 番，つまり SHIFT を押しながら F 5 を押すと，CONT RETURN が実行されるので利用するといいでしょう．



ただし注意しなければならないことがあります．中断しているときにプログラムリストをいじってしまうと，もう CONT 命令は使えなくなるのです．たとえばいまのプログラムで，どこかでプログラムを中断させ，

```
100 PRINT 2
```

と書き換えてから CONT を行ってみましょう．



```
?CN Error
```

と表示されて，CONT 命令が使えないことを示します．

6－3 途中からごめんください

GOTO 命令を使うと，好きな行番号のところへ移ることができますが，この命令をダイレクトモードで実行するとどうなるでしょうか．まず次のようなプログラムを考えてみましょう．

```
100 END
110 PRINT "A=";A
120 PRINT "B=";B
130 END
```

このプログラムは最初にEND命令があるため，RUNしても何も起こりませんね．ところがここでGOTO命令を使うと，プログラムの途中から実行を開始できるのです．ダイレクトモードで，




```
GOTO 110
```

と命令してみてください．110行と120行が実行されます．また，

```
GOTO 120
```

とすれば120行だけ実行されます．また，RUN命令で実行したときには，最初に変数はすべて0になってしまうのですが，GOTO命令ではそんなことはありません．

```
A=12:B=34:GOTO 110
```



これをダイレクトで行えば，変数の値がそのまま表示されるのがわかります．

実はRUN命令でも，プログラムの途中から実行することはできます．GOTO命令と同じように，後ろに行番号を書けばよいのです．ただしこの場合にもやはり変数はすべて0になってしまいます．次のように試してみてください．

```
A=12:B=34:RUN 110
```

6－4　あとの表示も考えて

普通PRINT命令が終わると最後に必ず改行が起こり，次のPRINT命令で表示される数や文字は，また新しい行の先頭に現れます．けれどもそれでは不便な場合も出てきます．たとえば画面を“あ”という文字で埋めつくすプログラムを考えてみましょう．

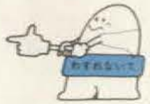
```
100 PRINT"ああああああああああああああああああああ
ああああああああああああ"
110 GOTO 100
```



100行では“あ”を32個表示させています．画面の1行は32文字なので，ちょうどいっぱいになり，それを繰り返せば画面が埋めつくされると考えたわけです．しかし実行してみればわかりますが，このプログラムでは1行おきの表示しかできないのです．なぜなら1行に32文字を表示するとカーソルは自動的に次の先頭に移ってしまうわけで，ここにPRINT命令が終わったときの改行が起こるため，まるまる1行分のすき間ができてしまうことになります．

100行を変えて“あ”を31文字にすれば行と行の間のすき間はできなくなりますが，こんどは画面の右端に1文字分のすき間が現れてしまい，目的の“画面を埋めつくす”ということはできません．

こういうときのため，PRINT命令の後に改行をさせないでおくことができます．それにはPRINT命令の最後にセミコロン（;）をつければよいのです．いまのプログラムの100行の最後にセミコロンをつけて試してみてください．



また，最後にセミコロンをつければ改行しないわけですから，実際には次のように書けば画面を埋めつくすことができます．このほうがずっと楽ですね．

```
100 PRINT "あ";
110 GOTO 100
```

セミコロンが使えるのなら，カンマ（,）はどうでしょうか．100行の最後をカンマに書き換えて実行してみてください．

```
100 PRINT "あ",
110 GOTO 100
```

こうすると，ひとつのPRINT文の中でカンマを使ったときとちょうど同じように，画面の1行に2つの割合で“あ”が表示されます．

結局カンマやセミコロンが，PRINT 文の最後につけられると，その動作は次の PRINT 文にまで影響してくるというわけです．



PRINT 命令の最後に何も書かないと改行が起こるというのも，使い方によっては役に立ちます．わざとここで改行させたいとか，ここに 1 行のすき間をあけたい，というときには，ただ

```
PRINT
```

と書いておけば，何も表示させないで改行だけ起こすことができます．次のプログラムでは，この方法をつかって結果を読みやすくしてみました．

```
100 INPUT "いま なん時";H
110 INPUT "    なん分";M
120 A=720-H*60-M
130 PRINT
140 PRINT "12時まで あと";A;" 分"
150 PRINT:PRINT
160 GOTO 100
```

ここで"結果を読みやすくする"ということが出てきましたが，これはとても大事なことですね．コンピュータを使ってせっかく便利な計算ができるようになっても，その答を読むのがたいへんなのでは，なんの意味もありません． 結果を読みやすくするためには，上のように行の間にすき間をあける他に，画面に余計な文字を表示しないという方法があります．つまり，クリアキーを押してから答を表示すればよいわけです．しかし PC が自分でクリアキーを押すことはできませんから，それと同じことをするための命令が用意されています．それが" CLS "命令です．上のプログラムに次のような行を追加して実行してみてください．



```
125 CLS
```

プログラムにこの CLR 命令があると，PC は画面をクリアし，カーソルをホームポジションへ戻してから次へ進むのです．

第7章　自動的に判断するプログラム



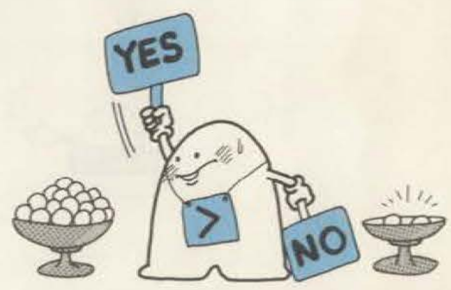
コンピュータが便利なのは，単に計算が速いとか，同じことを何回でも文句をいわずに実行するからだけではありません．人間が細かい指示や命令を与えなくても，自分で状況を見きわめ，それに従って動作を変えるという能力，つまりある程度の“考える”能力を持っているということもコンピュータの大きな魅力のひとつです．

この7章では，PCに“考え”させるためのための基本的な命令について説明しましょう．きみのプログラム次第では，PCが“人工知能”となる日もくるかもしれませんよ．

7－1　条件判断はＩＦ～ＴＨＥＮで

GOTO命令を使えば，PCに自動的に繰り返し作業をやらせることができました．けれどもその繰り返しをいつストップするかは，きみにまかされていたわけです．そこでこんどはそのストップの判断もPC自身にやらせてみましょう．たとえば次のプログラムを実行して，結果を見ていてください．

```
100 A=0
110 A=A+1
120 PRINT A
130 IF A>9 THEN END
140 GOTO 110
```



さっきの無限ループと同じように，1，2，3… と表示されはじめますが，こんどは10まで表示したところでプログラムが止まります．これは130行の命令が働いたのです．この130行の意味は“もしA>9ならENDを実行しなさい”ということです．この“もし～なら～する”という考え方は，

コンピュータが複雑な仕事をこなすためにどうしても必要なもので，“条件判断”の能力と呼ばれます．



PCのBASICで条件判断を行うのが，ここに出てきた“IF～THEN”です．プログラム中に“IF”があると，まずその後ろに書いてある条件を調べ，もし成り立っていれば“THEN”の後に続く命令を実行します．条件が成り立っていなければ，その行はすべて無視して，次の行の先頭からプログラムは実行されます．

前のプログラムの例では，“より大きい（>）”という条件が成り立つかどうか判断されたわけですが，他にもざっと次のような条件を判断することができます．



```
a>b     aはbより大きい
a<b     aはbより小さい
a=b     aはbと等しい
a>=b  ）aはbより大きいか等しい
a=>b
a<=b  ）aはbより小さいか等しい
a=<b
a<>b  ）aはbと等しくない
a><b
```

この中で，“＝”の条件判断については特別な注意が必要です．たとえば次のプログラムを実行してみてください．

```
100 A=0
110 A=A+0.1
120 PRINT A
130 IF A=1 THEN END
140 GOTO 110
```



このプログラムは，Aが1になったときに止まると思うでしょう．ところがやってみればわかるように，1で止まるどころか無限ループに入ってしまいます．それでは130行を，

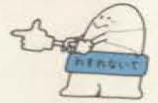
```
130 IF A>=1 THEN END
```

のように変えてから実行してみてください．こんどはちゃんと1で止まります．どうして“=”の条件判断ではいけないのでしょうか．

いま変数Aの値は1になっているはずですね．表示させてそのことを確かめてください．それでは，A−1を計算すると0になるわけですね．

```
? A-1
 4.65661287E-10
Ok
```

しかしこのとおり，A−1は0ではありません．どうやらAは1よりほんのわずかだけ大きいようです．つまりAは1と等しくないのです．だから最初に“=”で判断してプログラムを止めるのがうまくいかなかったのです．



コンピュータでは，小数点以下の数を扱うときには“誤差”が出てくるのが普通です．目には見えなくてもほんのわずか違っているということがありますから，等号“=”で等しいことを判判するのは間違いのもと，必ず“>=”などを使ってください．

IF～THENの条件が成立した場合に実行される命令は，PRINT文でも代入文でも何でもかまいません．また，GOTO命令が実行されるときに限って，次のような省略が可能です．

```
IF A=B THEN GOTO 100 ┐
IF A=B THEN 100      ├ これらは全て同じ意味
IF A=B GOTO 100      ┘
```



THENの後にマルチステートメントが続く場合は，条件が成立したときはそれらすべてが実行され，条件が成立しなかったときにはそれらすべてが無視されます．

```
IF A=B THEN X=1:Y=2:Z=3
```

a=bならこの3つが実行される．
a<>bなら，ひとつも実行されない

7－2　でたらめも使いよう



ゲームなどを考える場合，コンピュータのやることがいつでも簡単に予想されてしまっては，おもしろ味がありません．そのために BASIC に用意されているのが RND 関数です．これは，使うたびにでたらめな値（乱数）が得られるもので“乱数関数”ともいいます．

RND（X）は，引数 X を持っていますが，この X には次のような意味があります．



X>0	0≦RND(X)<1 の範囲の乱数
X=0	直前に使った乱数と同じ値
X<0	乱数系列を初期化した値

X>0 の場合，X はどんな数でも結果は変わりませんから，ここでは 1 を使いましょう．RND(1) は上に述べたように，0 以上 1 未満の乱数となります．次のプログラムを実行してみましょう．

```
100 PRINT RND(1)
110 GOTO 100
```



どうですか．でたらめな数がつぎつぎに表示されていくでしょう．そうしたらこんどは 100 行を次のように変えてみてください．

```
100 PRINT INT(RND(1)*6)
```

この場合は 0 から 5 までの 6 個の整数がでたらめな順序で表示されます．まず RND(1)＊6 で，0 以上 6 未満の範囲の乱数となり，その小数点以下を INT 関数で切り捨てるので，0 から 5 までの整数が得られるというわけです（この場合，6 という整数は絶対に得られないことに注意してください）．

この方法を使うと，BASIC でサイコロを振るまねをさせることができます．次のプログラムを実行してみましょう．“よござんすね？”と聞かれたらリターンキーを押してください．




```
100 INPUT "よこ゛さ゛んすね";A$
110 PRINT "はいります。 は゜っ!"
120 P=INT(RND(1)*6)+1
130 Q=INT(RND(1)*6)+1
140 PRINT P;" -";Q;" の";
150 R=P+Q
160 IF R-INT(R/2)*2=0 GOTO 180
170 PRINT "はん て゛す。":GOTO 100
180 PRINT "ちょう て゛す。":GOTO 100
```

120行と130行では，1から6までのでたらめな整数がPとQに代入されます．つまりサイコロを振るのと同じようなことをしているわけです．

次に130行のRND（1）をRND（0）と変えて実行してみるとどうなるでしょうか．この場合，2つのサイコロはいつでも同じ数になりますね．RND（0）というのは，そのすぐ前で使った乱数がもういちど得られるのです．そのため変数Qには変数Pと同じ数が代入されたのです．

さて，こんどは90行に次のものをつけ加えてみてください．

```
90 P=RND(-1)
```

こうすると，RUNした直後のサイコロは必ず5－5になるでしょう．その後には1－1，6－6，6－6と続くはずです．このようにRND(X)の引数Xを負の数にすると，乱数はある決まった値に設定され，以後は誰がやっても同じ並び方の乱数を得ることができます．引数Xの値によって最初に決まる値も変わってきますから，試しておいてください．

7－3　手作り関数はいかが

もういちど前に出てきたサイコロのプログラムを見てください．160行に“R－INT(R/2) ∗ 2”という式がありますね．この式は，Rを2で割った余りを求めるもので，Rが偶数なら答は0，Rが奇数なら答は1になります．

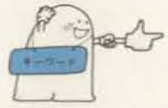
つまりRの値によって答が変わってくるわけで，ちょうどRを引数とした関数の一種だということもできます．BASICでは，これを実際に新しい関数としてつくり出すことができ，そうするとプ

ログラムもわかりやすくなります．それにはこのサイコロプログラムに，次のような変更を加えてください．




```
90 DEF FNA(X)=X-INT(X/2)*2
160 IF FNA(R)=0 GOTO 180
```

90行が新しく関数を定義している命令です．これは，“FNA (X)”という名前の関数を作ります．この命令は，“引数Xに対してX－INT (X/2)＊2を計算するものです”とBASICに教える働きを持っていて，“DEF FN 命令”と呼ばれます．



160行で“FNA (R)”という関数が出てくるとPCは90行の式を基にして，R－INT (R/2)＊2を計算することになります.このように，最初に関数を作り出すときに使った引数（この場合はX）と，実際使うときの引数（この場合はR）は，違っていてもいっこうにかまいません．それどころか実際に使う場合，カッコの中にさらに計算式が入ってもよいのです．たとえばこのプログラムではいま，

```
150 R=P+Q
160 IF FNA(R)=0 GOTO 180
```

のように，いったんP+Qを計算してから使っていますが，これは150行を消して，160行を

```
160 IF FNA(P+Q)=0 GOTO 180
```

と書いてもよいのです．FNA (X)が“Xを2で割った余り”を意味しているのですから，FNA (P+Q) は，“P+Qを2で割った余り”ということになるわけです．



新しく作り出す関数の名前は，必ず頭に“FN”の2文字をつけ，その後ろにそれぞれの関数を区別するための文字を続けます．この区別するための文字は，アルファベット1文字か、またはその後ろにアルファベットまたは数字がついたもので，つまり変数の名前をつけるときとまったく同じ規則があるということです．

7－4　FOR～NEXTはループの専門家

IF～THEN を使って繰り返しをするのもいいですが，最初から繰り返す回数が決まっているものは，その道の専門家にやってもらいましょう．

それが，FOR～NEXT です．次のプログラムを見てください．



```
100 A=1
110 FOR I=2 TO 10
120 A=A*I
130 PRINT A
140 NEXT I
150 END
```

これは，1から10までの数を順にかけて，それぞれの時点での結果を表示するものです．



まず，目につくのが，110行です．これが，FOR 命令の書き方です．FOR の次に代入命令がありますが，ここにある変数を，“ループ変数”とか，“ループ制御変数”といいます．そして代入されている値が，“初期値”です．代入命令の次に，TO というのがあって今度はただの数値があります．これが“終値”です．

FOR 命令による繰り返しを，単に“FOR ループ”ともいいますが，FOR ループでは，このようにループ変数に対して初期値から終値までの値をとりながら繰り返しをさせます．

そこで，どこまでを繰り返すのかを示すのが，NEXT の役目です．今のプログラムでは，140行に NEXT がありますから，その間の120行と130行を繰り返すわけです．NEXT 命令では，対応を示すためにループ変数を書いておきますが，これを省略することもできます．



さて，今のプログラムの流れを見てみましょう．まず，FOR 命令で I に 2 が代入されます．そして，120行，130行を実行して１４０行の NEXT に行き当たると I に 1 がたされます．それが終値を超えていなければ，FOR の次の命令に戻ります．今の場合，次の命令は次の行にありましたが，もし FOR 命令がマルチステートメントになっていたとしても大丈夫です．試しに次のように変更して実行してみてください．




```
110 PRINT "はじめは";A:FOR I=2 TO 1
0:PRINT "つぎは";
```

ただし，FORのある行をマルチステートメントにすると，プログラムがみにくくなりますから，なるべくなら，マルチステートメントにはしないでください．

さて，話を戻しましょう．Iが10になったときに，NEXT命令にさしかかるとIに1がたされて11になります．これは終値を超えていますから，そこで繰り返しはおしまいです．本当にそうなっているのか確かめましょう．次の行を追加してください．

```
145 PRINT I
```

実行すると，最後に11と表示されましたね．

ところで，Iは繰り返しの中で自動的に1ずつ増えましたが，それ以外の数を増やすことはできないのでしょうか．

もちろん，そんなことは簡単です．BASICでは，便利な命令はあくまでも便利なように作られているのです．では，110行を次のように変更してください．

```
110 FOR I=1 TO 2 STEP 0.1
```



終値の後に，“STEP”というのを付けました．この後に書かれた数値を“増分”といいます．そして，この場合はNEXTのところで0.1がたされることになります．では，実行してみてください．

おっと！　最後になぜか“2”だけが表示されましたね．これはFORループを終えたときのループ変数の値だったはずです．終値が2なのに，なぜこうなってしまったかは，IFのところでやったのと同じ理由です．つまり，この2は実は，2よりもほんのちょっと大きいからです．確かめてみましょう．もう1度実行してみて終わったところで，次のように打ちこんでください．

```
? I-2
 9.31322575E-10
```

どうです．2よりも100億分の9だけ大きいのです．増分に小数を使うときはこんなことがありますから注意してください．

さて，今までは，初期値，終値，増分ともに数値を使っていましたが，変数にすることもできます．次のプログラムを実行してみてください．

```
100 INPUT "しょきち";A
110 INPUT "しゅうち";B
120 INPUT "ぞうぶん";C
130 FOR I=A TO B STEP C
140 NEXT I
150 END
```

さて，FORループの中には何もありませんから，すぐに終わってしまうでしょうね．では，まず"しょきち"に1，"しゅうち"に10，"ぞうぶん"に1とそれぞれ答えてください．どうです．思ったとおりすぐに"Ok"が表示されましたね．

では，次に"しょきち"に1，"しゅうち"に10000，"ぞうぶん"に1と答えてください．おや，なかなか"Ok"が出ませんね．どうしたんでしょう．

実は，FORループの中に何もいれなくても，PCはインチキせずにループ変数に初期値を代入し，増分をたしこみ，終値を超えたかを調べているのです．今のだと，1から10000まで数を数えたのと同じです．いかにコンピュータが速いといえど，こんなに数を数えるには，それなりの時間がかかるのはあたりまえです．

そこで，BASICではこのように何もしないFORループを，速すぎて困るところに入れて，ちょうどいい速さにすることがあります．そのような使い方を"時間待ち"とか"ウエイトループ"といいます．もう1度プログラムを実行して，今度は"しょきち"に1500，"しゅうち"に－1500，"ぞうぶん"に1を答えてください．すぐに終わってしまいましたね．初期値より終値が小さければ，繰り返しはただの1度だけになるのは当然です．次の1行を追加して，もう1度今答えた値でやってみてください．




```
135 PRINT I
```

1500と表示されましたね．

しかし，ループ変数の値をだんだん小さくしていくようなFORループが必要なこともあります．そういうときには，増分に負の数を使えばいいのです．もう1度プログラムを実行して，今度は“しょきち”に15，“しゅうち”に−15，“ぞうぶん”に−1と答えてください．

速くてよくわかりませんが，15から1ずつ減った数が表示されたようですね．

さて，FORループを使ううえでいくつか注意しなければならないことがあります．

まず，ループ変数はFORとNEXTで同じものでなければなりません．たとえば，さっきのプログラムで，

```
140 NEXT K
```

となっていたら，本当のループ変数はIですから，次のようなエラーが出ます．




```
?NF Error in 140
```

これは，FORが無いのにNEXTが出てきたというエラーで，NEXTで示されたループ変数に対応するFOR命令が無かったために出たのです．

こんなに短いプログラムならひと目で何がどうなっているかがわかりますから，こんなエラーは出さないでしょう．でも，繰り返しの部分が長いプログラムになってくると，よく全然別の部分からFORループの繰り返しの部分へGOTOで飛んできてしまうことがありますから注意してください．

次に，ループの中でループ変数には代入してはならないということです．次のプログラムを実行してみてください．




```
100 FOR I=1 TO 100
110 PRINT I
120 I=50
130 NEXT I
140 END
```

最初に1回だけ，1が表示されて後はずっと51のままで，いつまでたっても止まらなくなってし

まいました．それもそのはず，NEXT でＩにいくら１をたしこんだところで，その前にＩが必ず50になっているからです．では，120行を次のように変更してください．

```
120 I=120
```

実行すると今度は１を表示して止まってしまいました．これも理由は，よくわかりますね．このように数値を代入した場合は簡単にわかりますが，次のように計算になると，どうなるかはちょっと考えてしまいます．

```
120 I=I*I
```

実行する前に，どうなるかまず考えてみてください．それから実行してみてください．どうです．考えたとおりになりましたか．

こんな短いプログラムなら，わざと代入してもわかりますが，もっと長いものになるとプログラムが理解しにくくなりますから，ループ変数へは代入してはいけません．



もし，どうしてもループ変数に代入したいときには，FOR ループの繰り返しは避けて，IF～THEN による繰り返しにしてください．

そして，最後はループからの脱出です．次のプログラムを見てください．

```
100 FOR I=1 TO 10
110 PRINT I
120 IF I>5 THEN 200
130 NEXT I
140 PRINT "ループ しゅうりょう"
150 END
200 PRINT "ループ だっしゅつ"
210 GOTO 150
```

ループ変数の値が５より大きくなったところで FOR ループを飛び出しています．実行すると，ちゃんと“ループ　だっしゅつ！”の表示がありますからループからの途中脱出はできます．

この極端な例として，NEXT のない FOR 命令があります．たとえば，今のプログラムの130行を取り除いてしまってください．どうです．これでも，PC はなんのエラーも出さずにプログラムを

実行しますね．

このように，FORループでは，FORのないNEXTはエラーになり，NEXTのないFORはエラーにならないことに注意してください．

7－5　ループの中のループ

FORループによる繰り返しは，注意事項さえ守れば非常に便利なものです．そのため，プログラムには必ず出てくるといっていいほど多く使われています．

そこで問題となってくるのが，ループの中にループがあるようなものです．次のプログラムをみてください．




```
100 FOR I=1 TO 14
110 FOR J=1 TO I*2
120 PRINT "*";
130 NEXT:PRINT:NEXT
140 END
```

これは，画面に簡単な模様を描くものですが，ループ変数Jによる内側のループと，Iによる外側のループがあるのがわかりますね．このように，ループの中にループを作ることを，ループの“ネスティング”といいます．

ネスティングをするときに，注意しなければならないことは，FORとNEXTの対応です．たとえば，130行のループ変数を省略してあるNEXT命令に，次のようにループ変数を対応してしまったらどうなるでしょう．

```
130 NEXT I:PRINT:NEXT J
```

さあ，実行してみてください．次のように表示されますね．

```
**************
?NF Error in 130
```

これでは，どちらのNEXT命令がエラーになったのかわかりませんから，この行を次のように2つに分けてみましょう．さあ，今度は，

```
130 NEXT I:PRINT
135 NEXT J
RUN
***************
?NF Error in 135
```

と出て，NEXT Jがエラーを引き起こしていたのがわかりました．これは，どういうことかというと，実はループ変数がJのループは，NEXT Iが出てきてしまった時点で消滅してしまっていて，110行はあってもなくても同じことになったからです．試しに，110行を取り除いて実行してみてください．

このように，FORループのネスティングは，FOR命令とは逆の順序でNEXT命令のループ変数を指定しなければならないことに注意してください．



次のプログラムです．

```
100 FOR I=1 TO 3
110 FOR J=1 TO I*2
120 PRINT I,J
130 NEXT:NEXT
140 END
```

このように，同じところにNEXT命令がある場合は，次のように書くこともできます．

```
130 NEXT J,I
```

この場合も，FORの出てきた順序とは逆の順に，ループ変数を並べなければならないのは当然です．

7－6 “いま”をとらえるINKEY$

プログラムを実行中にINPUT命令があると，PCはそこで止まってきみの入力を待ちます．しかし，いちいち止まって待たれたのでは困る場合もあります．たとえばテレビゲームのようなものでは，INPUT文で止まってしまっては使いものになりません．こういうときには，プログラムは止めないでPCに指示を与える方法があるといいですね．



それを実現するのが“INKEY$”です．これはその名前からわかるように，文字変数の一種です．ただし文字列を代入することはできません．それではどんな文字がこのINKEY$に入っているのかというと，ちょうどそのとき押されたキーの1文字が入るのです．キーが何も押されていなければ何もない文字，つまりヌルストリングが入ります．

次のプログラムを実行してください．これはINKEY$の内容をそのまま表示するだけのプログラムで，キーを押すとINKEY$がいったいどう変化するのか目で見ることができます．

```
100 PRINT "[";INKEY$;"]";
110 FOR I=1 TO 50:NEXT
120 GOTO 100
```



このプログラムを使っていろいろ試してみると，次のようなことがわかります．まず，キーを押していれば必ずその文字がINKEY$に入るのではなく，押した瞬間一回だけ，そのキーの文字が読まれます．キーを押したり離したりすると，そのたびにINKEY$に文字が入ります．少しキーを押し続けるとオートリピートが始まりますが，これはキーを押したり離したりしているのと同じで，どんどんキーは読みこまれます．実際はキーを押すとそれがすぐにINKEY$に入るのではありません．いったん押された順に記憶され，後からそれをひとつずつ取ってきてはINKEY$としています．そのため，プログラム中でINKEY$が出てくる速さよりキーを押すほうが速いと，押したキーがどんどんたまっていってしまいます．これはプログラムの110行の時間待ちループを，

```
110 FOR I=1 TO 200:NEXT
```

と変えてスピードダウンすると，もっとよくわかると思います．

第8章　データをいっぱい扱ってみよう



きみはすでに，第7章まで読み進んできたはずです．第7章のFOR文など，わかってもらえたでしょうか．わかってもらえた人には，BASICがどんな言語であるかがなんとなくつかめてきたことだと思います．

第8章では，データがいっぱいある場合に，N60-BASICでは配列という便利なものが使えることをお教えしましょう．この配列というものを使うと，プログラムがより簡単にできるんですよ！

8－1　配列は便利

今までに覚えた命令を使って，5つの数値のデータを入力して，それを大きい順に並べ換えるプログラムを考えてみましょう．

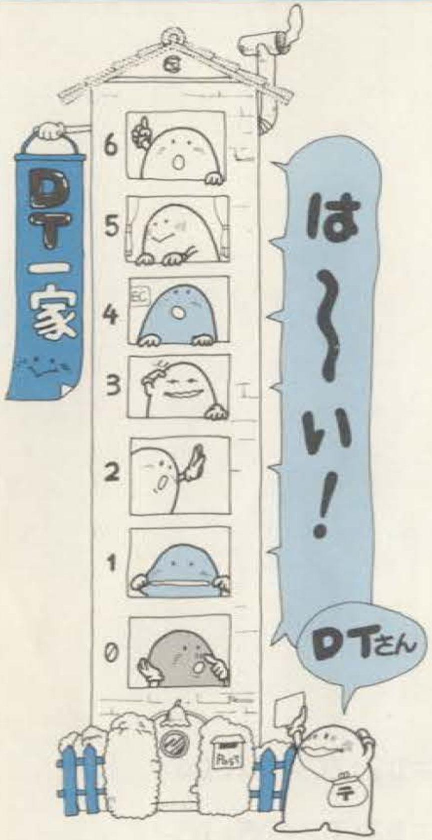
```
100 INPUT "Data 1 =";D1
110 INPUT "Data 2 =";D2
120 INPUT "Data 3 =";D3
130 INPUT "Data 4 =";D4
140 INPUT "Data 5 =";D5
150 IF D1<D2 THEN C=D1:D1=D2:D2=
C
160 IF D1<D3 THEN C=D1:D1=D3:D3=
C
170 IF D1<D4 THEN C=D1:D1=D4:D4=
C
180 IF D1<D5 THEN C=D1:D1=D5:D5=
C
190 IF D2<D3 THEN C=D2:D2=D3:D3=
C
200 IF D2<D4 THEN C=D2:D2=D4:D4=
C
210 IF D2<D5 THEN C=D2:D2=D5:D5=
C
220 IF D3<D4 THEN C=D3:D3=D4:D4=
C
230 IF D3<D5 THEN C=D3:D3=D5:D5=
C
240 IF D4<D5 THEN C=D4:D4=D5:D5=
C
250 PRINT "せいれつ けっか"
260 PRINT D1;" ";D2;" ";D3;" ";D
4;" ";D5
270 END
```



プログラム続く

多分，このようになると思います．なんだか同じような命令がいっぱい並んでいますね．でも，データの個数が増えた時には大変です．

では，次のプログラムを入れてみてください．




```
100 DIM DT(5)
110 FOR I=1 TO 5
120 PRINT "Data";I;" =";:INPUT D
T(I)
130 NEXT I
140 FOR I=1 TO 4
150 FOR J=I+1 TO 5
160 IF DT(I)<DT(J) THEN C=DT(I):
DT(I)=DT(J):DT(J)=C
170 NEXT J,I
180 PRINT "せいれつ けっか"
190 FOR I=1 TO 5
200 PRINT DT(I);" ";
210 NEXT I
220 END
```

おや，前のプログラムと違って同じ命令が並んでいませんね．そうなんです．配列というものを使うと，こんな感じのプログラムになるんです．この場合はデータが5つでしたが，このプログラムは少し直すだけで，10でも20でもデータを並べ換えることができるんですよ！

では，プログラムの説明をしましょう．



まず，DIM DT(5) という目新しい命令がありますね．これは，DT(0)，DT(1)，DT(2)，DT(3)，DT(4)，DT(5) という6つの"配列変数"を使うことをあらかじめPCに教えてあげるもので，このことを"配列を宣言する"と言います．この宣言をされると，PCはその配列変数に必要なだけの記憶場所を用意してくれるのです．

たとえば，A(n) を宣言すると，A(0) から A(n) までのn+1コの配列変数が使えるようになるわけです．ここで，配列変数のかっこの中の数字を"配列の添字"ということも覚えておいてください．



賢いPCはそれだけではなく，この宣言をしなくても添字が10までの配列変数が現れると，自動的に内部で宣言をしてくれます．しかし，すでに宣言された(自動的に宣言されたものも含む)“配列名”(このプログラムではDT)をもう1度宣言しようとすると，たとえば，

```
100 DT(1)=1
110 DIM DT(10)
120 END
```

では，

```
?DD Error in 110
```



となり，2度は宣言できませんよといってPCに怒られます．自分で宣言をした配列ならまだしもPCが自動的に宣言した配列で怒られたのではたまりませんから，なるべく自分で宣言をするようにしたほうがいいですよ．

ところで配列名のことですが，これも普通の変数と同じ様に先頭の2文字までしかPCは識別してくれないので，このことも覚えておいてください．



プログラムを見ていくと，DT(I)というものがありますね．実はこれが配列の最大の長所なのです．何が長所かといいますと，Iという変数の値を変えると，なんとDT(I)という配列変数がさっきとは別の変数になるんです．つまり，I=1の時にはDT(1)であったものが，I=5ではDT(5)という変数に変わってしまうわけです．つまりこのプログラムではDT(I)の形でDT(0)からDT(5)までの配列変数が表せるんです．

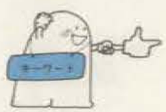
ここで，DT(I)という書き方が関数の書き方と似ていると思った人がいるかもしれませんが，これは使い方が違うので注意してください．

ところで，配列の宣言の所でDIM　DT(5)と宣言して，0から5以外の添字を使おうとしたらどうなるでしょう．

次のプログラムを実行してみてください．

```
100 DIM DT(5)
110 DT(0)=1.5
120 PRINT "DT(0)=";DT(0)
130 DT(5)=1.7
140 PRINT "DT(5)=";DT(5)
150 DT(6)=1.2
160 PRINT "DT(6)=";DT(6)
170 END
```

添字が0から5までの場合は問題がないようですが，DT(6) に数を代入しようとした所で，



?BS Error

となり，添字が正しくないですよと怒られました．つまり，宣言をしていない添字の配列変数は使えないということなんです．

ここまでの話で，だいたい配列についてはわかってもらえたと思います．しかし，配列にはまだまだ使い方があります．

配列というものは，配列による整列のプログラムのように，FOR ループを用いてそのループ変数を添字として扱います．

今までは数を扱う配列だけについて話してきました．しかし，ちょっと思い出してください．普通の変数には，数を扱う変数のほかに文字変数というものがありました．そうです，配列にも“文字配列”があるんです．そして，普通の変数と文字変数の使い方の違いと，普通の配列と文字配列の使い方の違いは同じなんです．つまり，配列名のあとに“$(ドル)”を付けたものが文字配列となるわけです．でも，扱うのが文字列であること以外，配列名や添字の規則はまったく同じです．

次に文字配列を用いたプログラムをあげておきますから，文字変数の場合と比べてよく見ておいてください．

```
100 DIM MOJI$(6)
110 MOJI$(0)="Sun"
120 MOJI$(1)="Mon"
```

プログラム続く



```
130 MOJI$(2)="Tues"
140 MOJI$(3)="Wednes"
150 MOJI$(4)="Thurs"
160 MOJI$(5)="Fri"
170 MOJI$(6)="Satur"
180 FOR I=0 TO 6
190 PRINT MOJI$(I)+"day"
200 NEXT I
210 END
```

あなたは今，学校の教室にいます．教室には図のように5行4列の机が並んでいて，座席に座っている人はみんなゼッケンをつけています．3行4列の人ゼッケンは12番で，5行2列の人は18番であることはわかりますね？



ここで次のプログラムを入れてみてください．

```
100 DIM NUM(5,4)
110 FOR I=1 TO 5
120 FOR J=1 TO 4
130 NUM(I,J)=(I-1)*4+J
140 NEXT J,I
150 INPUT "なんき゛ょうめ て゛すか";I
160 IF I=0 THEN END
170 INPUT "なんれつめ て゛すか";J
180 PRINT I;"き゛ょう";J;"れつ の ひと は"
;NUM(I,J);"は゛ん て゛す"
190 GOTO 150
```



おや？　と思われた人もいると思いますが，配列の添字が2つになっています．そうです．配列では**複数の添字**でも書けるのです．ですから，縦横に広がるこの例のような場合には，添字を2つ使えばプログラムも書きやすくなるでしょう．

これは，文字配列の場合も同じです．

8－2　いろんなデータでやってみよう！

	1列↓	2列↓	3列↓	4列↓
1行→	11	12	22	9
2行→	2	24	14	7
3行→	20	4	1	16
4行→	21	25	26	5
5行→	6	15	30	10

前の例ではゼッケンの番号が順番に並んでいましたが，これらが順番になっていなければ，前のプログラムのように行と列の数（IとJ）から計算して，ゼッケンの番号として配列に入れてやるということができません．

では，左の図を見てください．

前の例と同じ配置の机ですが，ゼッケンの番号がバラバラになっています．

ここで，次のプログラムを入れてみてください．

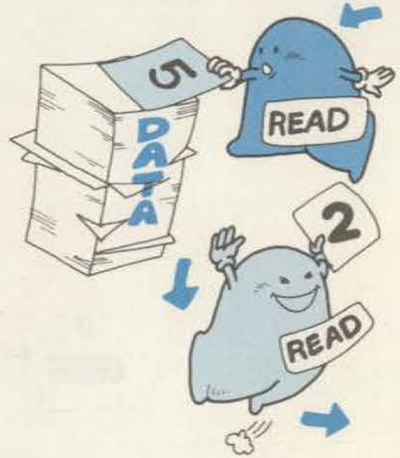
```
100 DIM NUM(5,4)
110 FOR I=1 TO 5
120 FOR J=1 TO 4
130 READ NUM(I,J)
140 NEXT J,I
150 INPUT "なんぎょうめ ですか";I
160 IF I=0 THEN END
170 INPUT "なんれつめ ですか";J
180 PRINT I;"ぎょう";J;"れつ の ひと は"
;NUM(I,J);"ばん です"
190 GOTO 150
200 DATA 11,12,22, 9
210 DATA  2,24,14, 7
220 DATA 20, 4, 1,16
230 DATA 21,25,26, 5
240 DATA  6,15,30,10
```

このプログラムを実行すると，指定した行と列の机に座っている人のゼッケンの番号がちゃんと出てくるでしょう．



このプログラムで用いているのが，READ文とDATA文です．READ文とDATA文は常に対となる命令で，READ文はそこに書かれている変数（配列変数も含む）にDATA文で書かれているデータを代入していくのです．DATA文にカンマ(,)で区切られて書かれているデータは，READ文が実行されるごとに1つずつ順に読まれていくわけです．

つまりこのプログラムでは，NUM(1, 1) には 11, NUM(1, 2) には12, ……NUM(5, 4) には18 が代入されるのです．ここで，READ 文には 1 度に複数の変数を書くこともできます．



次のプログラムを入れてみてください．

```
100 READ A,B,C
110 PRINT A;B;C
120 GOTO 100
130 DATA 1,2.5,3.14159265
140 DATA 6
150 DATA 8.3,0.4
```

このプログラムを実行すると

```
1 2.5 3.14159265
6 8.3 .4
```

と表示後に

```
?OD Error
```



となり，データがたりませんよと PC がいってきます．これは，2度までは READ 文を正常に実行して，3度目に READ 文を実行しようとしたところで読むべきデータがなくなってしまったためです．ですから，READ 文で読もうとするデータの個数と DATA 文で書いておくデータの個数は，必ず合わせておかなければなりません．

DATA 文のデータの個数が多い分にはかまわないのですが，それでも余分には書いておかない方がよいでしょう．DATA 文のことですが，これは実際にこれ自体実行される命令ではないので，プログラム中のどこにあってもいいという，特殊な命令です．しかし，プログラムの最後などに，なるべく 1 カ所にまとめておくことをお勧めします．

今までの READ 文，DATA 文の使い方ではデータも最初から順に読んでいくだけなのですが，全く同じデータをもう一度読みたいときには，また同じデータを書かなければならないので不便です．

次のプログラムを入れてみてください．

```
100 FOR I=1 TO 3
110 PRINT I;"かいめ"
120 FOR J=1 TO 3
130 READ A,B
140 PRINT A;B
150 NEXT J
160 RESTORE
170 NEXT I:END
180 DATA 1,2,3,4,5,6
```



このプログラムを実行すればわかると思いますが，RESTORE 文を用いるとデータをまた最初から読み直してくれるわけです．

ここで，RESTORE 文にはもう１つの使い方があります．

次のプログラムを入れてみてください．

```
100 FOR I=1 TO 2
110 FOR J=1 TO 3
120 READ A,B
130 PRINT A;B
140 NEXT J
150 PRINT:RESTORE 180
160 NEXT I:END
170 DATA 1,2
180 DATA 3,4
190 DATA 5,6
200 DATA 7,8
```

このプログラムのように，RESTORE 文の後に行番号を書くと，その行番号かそれ以後にある DATA 文からデータを読み始めてくれるわけです．しかしここで，存在しない行番号を書くと，



```
?UL Error in 150
```

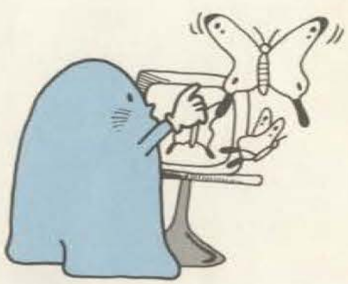
となり，そんな行番号は存在しませんよと PC に怒られるので注意してくださいね．ただこの行番号のところには変数は使えませんので，あしからず．

READ 文, DATA 文では配列と同様に文字配列が使えます. しかし, 少し注意が必要です. 次のプログラムを入れてみて, 実行する前に自分でどのような結果がでてくるか考えておいてください.



```
100 DIM MOJI$(2,2)
110 FOR I=0 TO 2
120 FOR J=0 TO 2
130 READ MOJI$(I,J)
140 PRINT "MOJI$(";I;",";J;")=#"
;MOJI$(I,J);"#"
150 NEXT J,I:END
160 DATA NEC,PC-6001
170 DATA "NEC,PC-6001"
180 DATA "N60:BASIC"
190 DATA 012;345:6789
200 DATA
210 DATA エヌイーシー," ",9'16"25
```

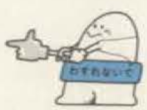
きみの考えていた答と実行結果が全く同じだったでしょうか. それはともかく, プログラムを説明しましょう.



プログラムとしてはMOJI$の文字配列にDATA文に書かれた文字列を代入し, これを, "#(ナンバー)"ではさんで表示するだけのものですが, これを見れば文字列がどのように代入されたかがよくわかると思います.

それでは, DATA文の書き方についてお話ししましょう. まず, "," や ":" を含まない文字列は "," で区切ればそのままデータとして書くことができます. また, "," や ":" を含む文字列については, その文字列の前後に " " " を付けることで代入されるようになります. 空白だけの文字列の場合も " " " を前後に付けて " "   " " のようにしてください.

最後に " " " を含む文字列についてですが, " " " が2文字目以降にある文字列は先頭に " " " を付けないようにすればそのまま代入することができます. しかし, " " "1文字だけや1文字目が " " " である文字列は, DATA文では " " " を代入することができません. これは, 先頭に " " " のある文字列と, 文字列の前後を " " " でくくったデータとの区別ができないからです.



付け加えておきますが，DATA文で“：”を区別している理由は，マルチステートメントができるようにするためで，その行を実行すると，マルチステートメントをちゃんと実行してくれます．また，空のDATA文（DATAの後に何もデータを書かない）があると，その文字変数にはヌルストリングが入ることになり，これはDATA文中で“,,”としたときと同じことになります．



きみはもう知っているかもしれませんが，DATA文中に書く文字列は小文字は大文字には変換されませんから，大文字にしたい時には大文字で入力してください．

最後にもう1つ．

次のプログラムを実行してみてください．

```
100 READ A$
110 PRINT "A$=";A$
120 READ A
130 PRINT "A=";A
140 READ B$
150 PRINT "B$=";B$
160 READ B
170 PRINT "B=";B
180 END
190 DATA 123,123
200 DATA NEC,NEC
```

変数Bに文字列“NEC”を代入する所で，




```
?SN Error in 200
```

となります．つまりBは普通の変数であるので，文字列は代入できないわけです．ところが，A及びA$では見た目には全く同じ数“123”を読ませていて，エラーが出ていません．これは，変数Aには数としての123が代入され，文字変数A$には文字列“123”が代入されたわけです．DATA文では，数値と文字は混在できるのですが，思いがけないデータが代入されないように，充分気をつけてプログラムを作ってくださいね！

第9章　同じような仕事は1ヵ所で



いよいよ第9章です．きみはもう8つの章を読み，そしてPCでいろいろ試してきたわけですが，もうPC-6001には馴れましたか．

この章では，プログラムの中のあちこちに出てくる同じ仕事を1ヵ所にまとめて，めんどうみようという方法を紹介します．

今度も，PCを本の横に置いて（本の下でも，本の前でもとにかく手の届くところに置いて），実際にいろいろ試してくださいね．

9－1　同じような仕事とは

PC-6001に限りませんが，コンピュータに計算やゲームをやらせるためには，人間がプログラムを作り，コンピュータに入力してやる必要がありましたね．

そこで，きみが何かのプログラムを作って，PCに打ちこむとしましょう．きみはキーボードからプログラムを打ちこむのは好きですか．プログラムが長ければ長いほど好き，ですか？

たぶん，キーを打つときは，10本もある両手の指の1本か2本しか使わず，次に打つべき文字を探しながら打つ人が多いのではないでしょうか．そんな人にとっては，プログラムは短ければ短いほど良いはずです．短ければ，打ち間違いが減るし，第1，手早くすんで早くゲームや計算ができるわけです．

もし，きみがタイプの名人でキーボードを見ずに両手を使って，プログラムを打ちこむことができたとしても，やはりプログラムは短い方がいいはずですよね．

この，プログラムをできるだけ短くする，ということのために，いろいろなテクニックが考えられてきました．たとえば，第7章のFORループを使えば，短いプログラムで何百回，何千回と同

じことをすることができます．

そこで，プログラムのあちこちにある同じような仕事をひとまとめにしておいて，必要なときにそれを使えるようにできたら便利だと思いませんか．そうすれば，全体のプログラムを短くすることができ，また同じようなことを何回も打ちこむ手間が省けます．その上，プログラムがすっきりするために，作りやすく，またわかりやすくなります．



そのために考えられたのが，“サブルーチン”というものです．“サブ”には，“副”といった意味があります．これに対する“主”は，“メイン”といいます．そして，“ルーチン”は，プログラムの一部で，ひとまとまりになったもの，という意味です．ですから，サブルーチンのことを副プログラムということもあります．



さて，サブルーチンに対して，これを使う方のプログラムを“メインルーチン”といいます．そして，サブルーチンを使うことを，サブルーチンを“コール”するといいます．

9－2　ＢＡＳＩＣのサブルーチン

では，さっそく次のプログラムを実行してみてください．

```
100 X=10
110 GOSUB 200
120 PRINT X
130 END
200 X=X*2
210 RETURN
```



さて，まず目につくのがGOSUBという命令ですね．それにプログラムの終わりを示すはずのEND命令の後ろにもプログラムが続いています．

実行させると，どうなりましたか．画面には，20と表示されましたね．変数Xには10を代入しているのに，Xの値を表示させると10ではなくて，20になってしまいました．原因は，この間にある，GOSUBという命令にありそうです．それでは，この行を取り除いてしまったらどうなるでしょう．

この110行を消すには，110に続いて RETURN キーを押すのでしたね．さて，実行させると今度

は，10と表示されました．つまり，Xは元の値のままです．やはり，秘密はGOSUB命令にあるようです．

それでは，110行を元のように入れなおして，200行を次のように変えてみてください．

```
200 X=X/2
```

実行させると今度は5と表示されましたね．もうきみには，GOSUB命令の意味が，なんとなくわかってきたでしょう．前にやったGOTOとよく似ているけれど，ちょっと違うみたいですね．試しに，GOSUBをGOTOに変えてみましょう．プログラムは次のようになりますね．

```
100 X=10
110 GOTO 200
120 PRINT X
130 END
200 X=X/2
210 RETURN
```

やってみてください．“ピッ”と音がして，

```
?RG Error in 210
```



が表示されましたね．どうやら，GOTOはGOSUBの代わりに使うことはできないようです．

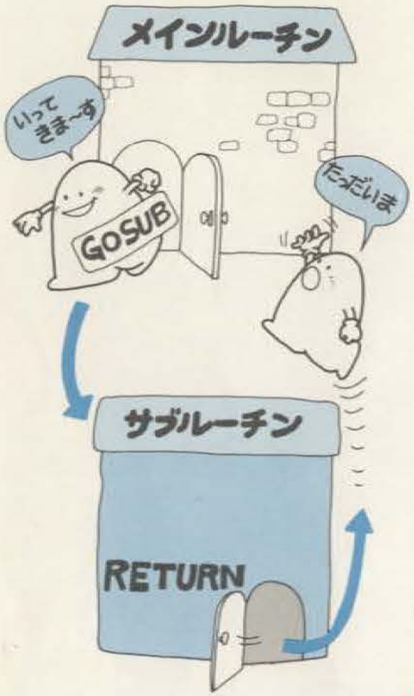
これでわかりましたね．GOSUB命令はサブルーチンをコールするもので，サブルーチンのありかはGOSUBに続く行番号で示されていたわけです．そして，サブルーチンの終わりのRETURN命令でコールされたところへ戻るわけです．ですから，このエラーメッセージの意味はコールされていないのに，RETURN命令が出てきたということになります．



ですから，このプログラムのサブルーチンは，最後の2行だけですが，これでも立派なルーチンなのです．このように，サブルーチンの終わりは必ずRETURN命令でなければなりません．

ではもし，このRETURN命令が無かったらどうなるでしょう．では，210行を取り除いてしまって，実行してみてください．

どうです．今度は何も表示されずに，Okが出てしまいましたね．この場合には，サブルーチンを



コールした後，そのままプログラムの最後になってしまったからですね．もし，この後にプログラムがあれば，そのプログラムを実行してしまいます．

では，サブルーチンの中でRETURNに出会うとどこへ戻るのでしょう．今までのプログラムでは，次の行へ戻っていましたが，もしGOSUB命令のある行が，マルチステートメントになっていたとしたらどうなるでしょう．

次のように，プログラムを変更してみましょう．

```
100 X=10
110 GOSUB 200:PRINT "X=";
120 PRINT X
130 END
200 X=X/2
210 RETURN
```

実行させると，ちゃんと，

```
X= 5
```

と表示されましたね．サブルーチンからは，GOSUBの次の命令へ戻ってくるわけです．

また，RETURN命令の後にマルチステートメントを使って別の命令を書いても，RETURN命令でメインルーチンへ戻ってしまいますから，その後の命令は実行されません．たとえば，次のプログラムを実行させると，何が表示されるでしょう．




```
100 X=10
110 GOSUB 200
120 PRINT X
130 END
200 RETURN:X=X/2
```

もちろん，5です．というのはうそですね．10のはずですよ．

ところで，今までのプログラムでは，メインルーチンからサブルーチンを1回しかコールしていませんでした．そのため，プログラムが短くなり，すっきりしてわかりやすくなる，といったサブ

ルーチンの利点は充分に生かされていませんでした．

では，次のプログラムはどうでしょう．



```
100 FOR I=1 TO 5
110 PRINT "Data";I;" =";:INPUT D
(I)
120 NEXT I:CLS
130 GOSUB 200
140 FOR I=1 TO 4
150 FOR J=I+1 TO 5
160 IF D(I)<D(J) THEN C=D(I):D(I
)=D(J):D(J)=C
170 NEXT J,I
180 PRINT "せいれつ けっか"
190 GOSUB 200:END
200 FOR I=1 TO 5
210 PRINT "Data ";I;" =";D(I)
220 NEXT I
230 RETURN
```

これは，前の章の数を大きい順に並べ換える整列プログラムです．そこに配列を表示するためのサブルーチンを付けて，入力した数値を確かめるためと，整列の結果を表示するために，メインルーチンの2ヵ所からコールしています．

もし，このプログラムをサブルーチンを使わなければ次のようになってしまいます．

```
100 FOR I=1 TO 5
110 PRINT "Data";I;" =";:INPUT D
(I)
120 NEXT I:CLS
130 FOR I=1 TO 5
133 PRINT "Data ";I;" =";D(I)
137 NEXT I
140 FOR I=1 TO 4
150 FOR J=I+1 TO 5
160 IF D(I)<D(J) THEN C=D(I):D(I
)=D(J):D(J)=C
170 NEXT J,I
180 PRINT "せいれつ けっか"
190 FOR I=1 TO 5
200 PRINT "Data ";I;" =";D(I)
210 NEXT I
220 END
```

プログラム続く



どうです．サブルーチンを使った前のプログラムの方が短いですね．

このようにサブルーチンを使うとプログラムが短くなり，その割合は，サブルーチン自体が長いほど，またコールするか所が多いほど，大きく違っていくことは簡単にわかりますね．

しかし，もう1つの見やすいという点では，今ひとつ納得がいきませんね．この程度の長さのプログラムならまだしも，もっと長いプログラムになったら，流れを追っていくだけでもたいへんなことになりそうです．

そうならないためには，どうすれば良いのでしょうか．たとえば，プログラムの実行には関係ないような，メモのような覚え書きがリストに出されれば，GOSUB の後にどんなサブルーチンをコールしたかわかるし，サブルーチンの最初の方はここからこういう仕事をするサブルーチンですよと書いておけます．



BASIC には，そのような覚え書きをするための命令が，しっかり用意されています．それが，REM 命令です．

REM 命令は，その命令以降行末までは何を書いても，BASIC は無視して次の行へ行けと指示します．ですから，

```
100 REM はいれつ ひょうじ゛:GOSUB 200
```

では，サブルーチンをコールできません．これは，

```
100 GOSUB 200:REM はいれつ ひょうじ゛
```

のようにしなければなりません．

そこで，さきほどのプログラムに適当にこの覚え書きを入れて，見やすいプログラムを作ってみましょう．次のようになりました．

```
100 REM かず゛ の せいれつ
105 FOR I=1 TO 5
110 PRINT "Data";I;" =";:INPUT D
(I)
120 NEXT I:CLS
```


プログラム続く



```
130 GOSUB 200:REM はいれつ ひょうじ
140 FOR I=1 TO 4
150 FOR J=I+1 TO 5
160 IF D(I)<D(J) THEN C=D(I):D(I
)=D(J):D(J)=C
170 NEXT J,I
180 PRINT "せいれつ けっか"
190 GOSUB 200:END
200 REM はいれつ の ひょうじ
210 FOR I=1 TO 5
220 PRINT "Data ";I;" =";D(I)
230 NEXT I
240 RETURN
```

どうです．ずっと見やすくて，わかりやすいプログラムになりましたね．これで，同じような仕事はサブルーチンの形にした方がよい，ということがわかりましたね．そして，サブルーチンは1つのプログラムにいくつあってもいい，ということもわかりますね．たとえば，



```
100 INPUT "X= ";X
110 INPUT "Y= ";Y
120 GOSUB 200
130 GOSUB 300
140 END
200 REM たしざ゛ん
210 PRINT "X+Y=";X+Y
220 RETURN
300 REM かけざ゛ん
310 PRINT "X*Y=";X*Y
320 RETURN
```

このプログラムには，2つのサブルーチンが使われています．

さて， に問題となるのは，サブルーチンはプログラムのどの辺にあるべきかということです．まさか，メインルーチンの真中にサブルーチンを書く人はいないでしょう．

サブルーチンは，それ自体で1つのまとまったプログラムですから,メインルーチンの前か後ろか

が問題なのです．では，次のプログラムをそれぞれ実行してみて，それぞれ“スタート”が表示されてから，“エンド”が表示されるまでの時間を計ってみてください．

```
100 GOTO 300:REM ﾒｲﾝﾙｰﾁﾝ ﾍ
200 REM ｻﾌﾞﾙｰﾁﾝ
210 RETURN
300 REM ﾒｲﾝﾙｰﾁﾝ
310 PRINT "ｽﾀｰﾄ"
320 FOR I=1 TO 10000
330 GOSUB 200
340 NEXT I
350 PRINT "ｴﾝﾄﾞ"
360 END
```

```
100 REM ﾒｲﾝﾙｰﾁﾝ
110 PRINT "ｽﾀｰﾄ"
120 FOR I=1 TO 10000
130 GOSUB 200
140 NEXT I
150 PRINT "ｴﾝﾄﾞ"
160 END
200 REM ｻﾌﾞﾙｰﾁﾝ
210 RETURN
```

どちらのプログラムが速かったですか．前の方ですね．それでは，サブルーチンはなんでも，メインルーチンの前の方へ書いた方がいいのかというと，実はそうはいきません．



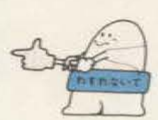
前のほうに長いサブルーチンがあれば，当然メインルーチンはどんどん後ろの方にいってしまいます．すると，今度はメインルーチン自体の実行する速さが遅くなってしまいます．そこで，一般的には，速くする必要のあるサブルーチン，またはよく使われるものだけはメインルーチンの前へ，そうでないものは後ろへもっていきます．

9－3　サブルーチンのサブルーチン

サブルーチンは便利だということはわかりましたね．便利なものは，どんどん使いましょう．

さて，ここで第7章でやったFORループのことを思い出してください．FORループの中で，さらにFORループを使うことをネスティングといいましたね．

サブルーチンの場合も，FORループのようなネスティングがあります．どんなものかというと，メインルーチンからコールされたサブルーチンが，さらに別のサブルーチンをコールするわけです．もちろん，この最後にコールされたサブルーチンが，さらに別のサブルーチンをコールすることもあるわけです．



このように，サブルーチンをネスティングしても，GOSUBとRETURNを正しく使っている限り，正しくプログラムを実行されます．

では，次のプログラムをみてください．

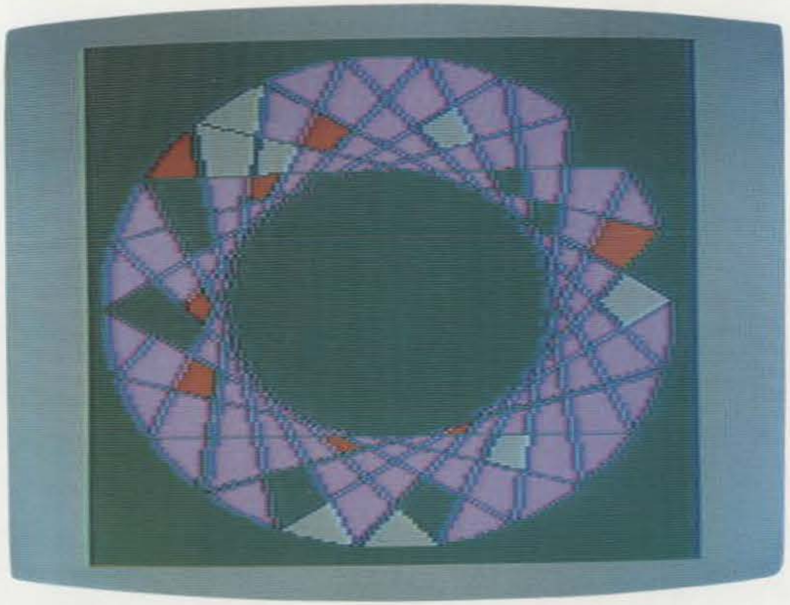
```
100 REM ﾒｲﾝﾙｰﾁﾝ
110 GOSUB 200
120 END
200 REM ｻﾌﾞﾙｰﾁﾝ 1
210 GOSUB 300
220 RETURN
300 REM ｻﾌﾞﾙｰﾁﾝ 2
310 GOSUB 200
320 RETURN
```

さあ，実行する前にどうなるか，考えてみましょう．まず，メインルーチンはサブルーチン1をコールして戻ってくると，ENDで終わりになります．次にサブルーチン1は，サブルーチン2をコールして戻ってくると，RETURNでコールされた所に戻ります．そして，サブルーチン2では，サブルーチン1をコールしてそしてRETURNで戻っています．

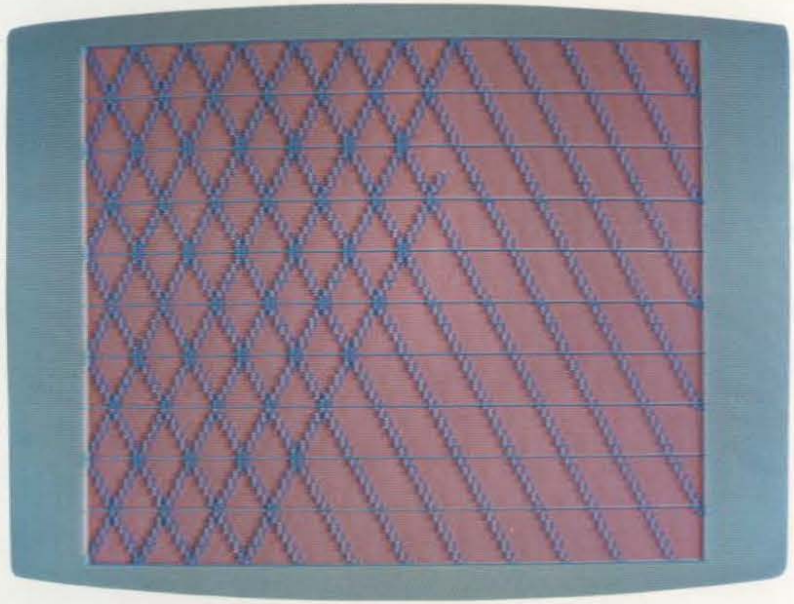
では，実行してみましょう．どうなりました．しばらくして，

```
?OM Error in 210
```



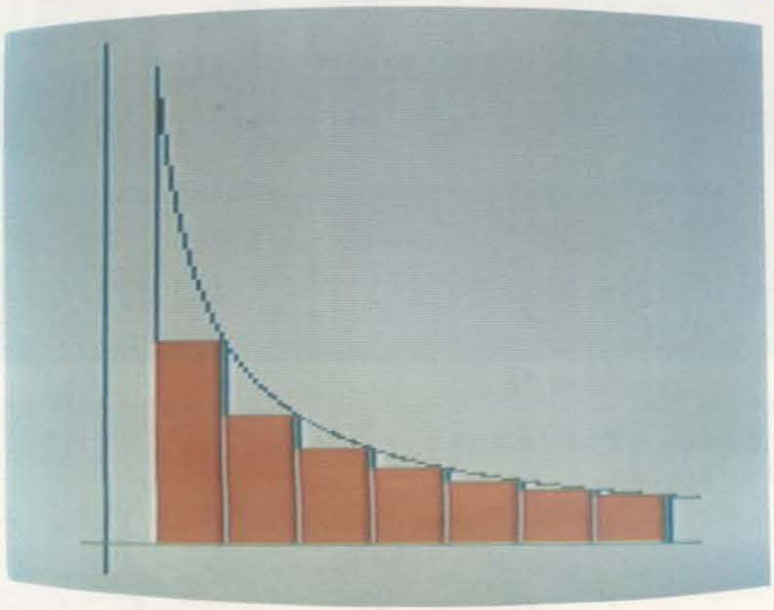


よしよし，おじさんが，こんなに大きなダイアモンドを買ってあげよう．

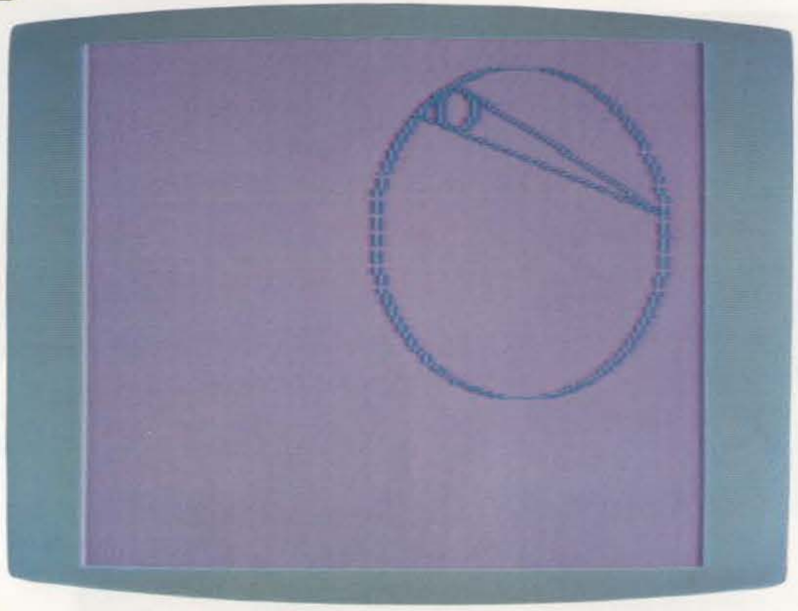


横に行きま～す．上に行きま～す．下に行きま～す．
三角形がいっぱいで～す．

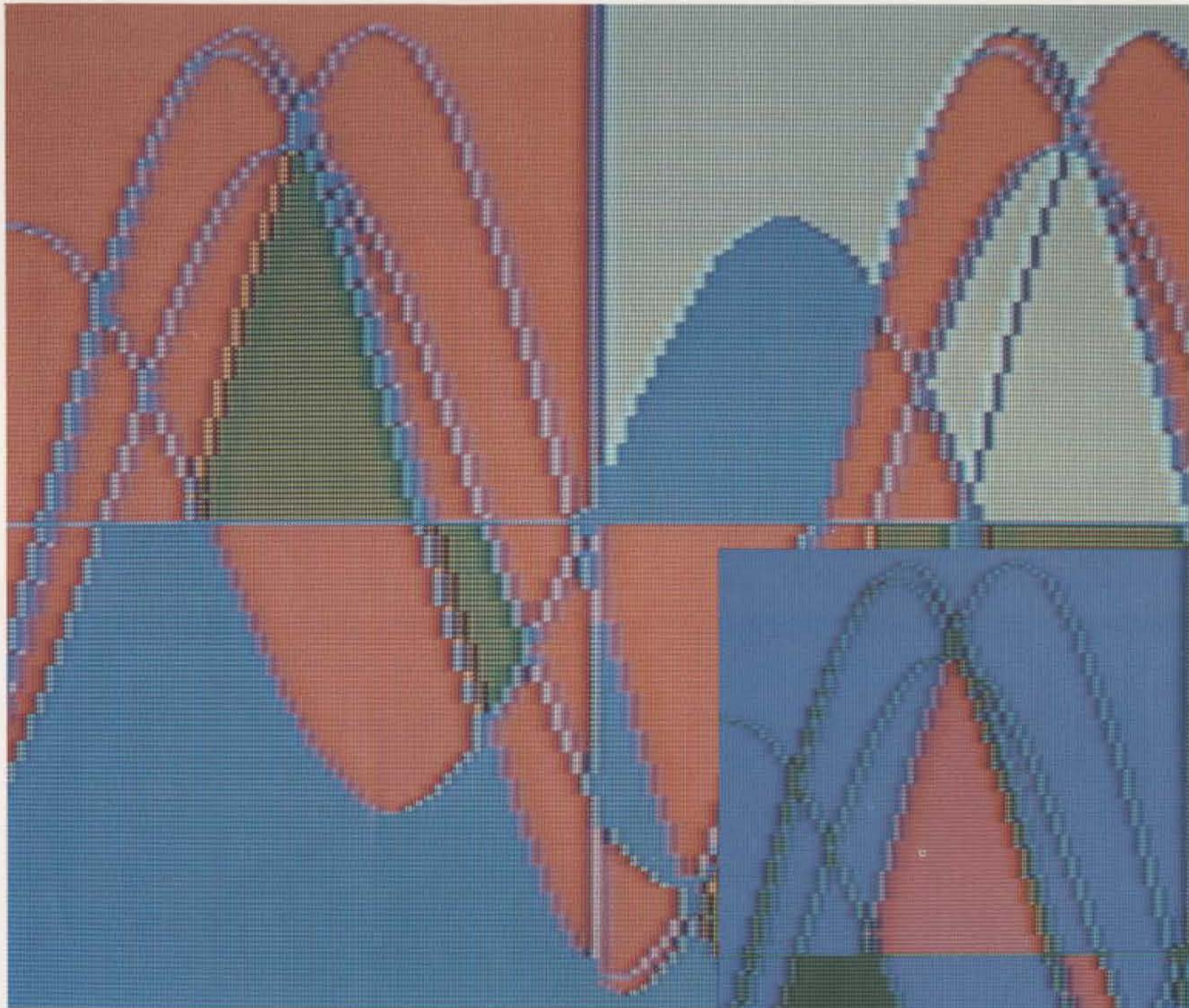


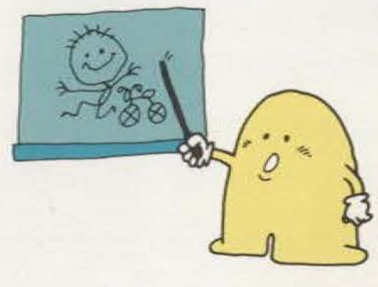


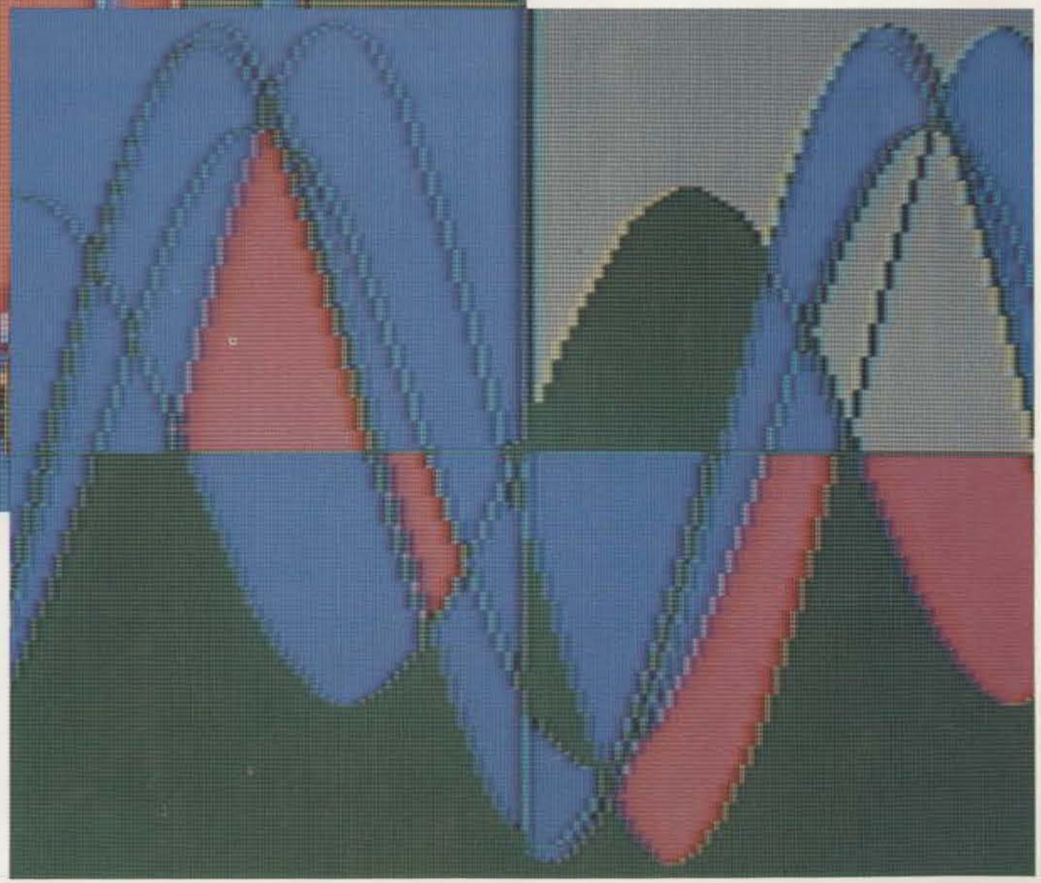
今月の成績は……？　あ～あ，また下がっちゃった！



ピンクの画面にあらわれた円．その中には三角形．またその中には…….

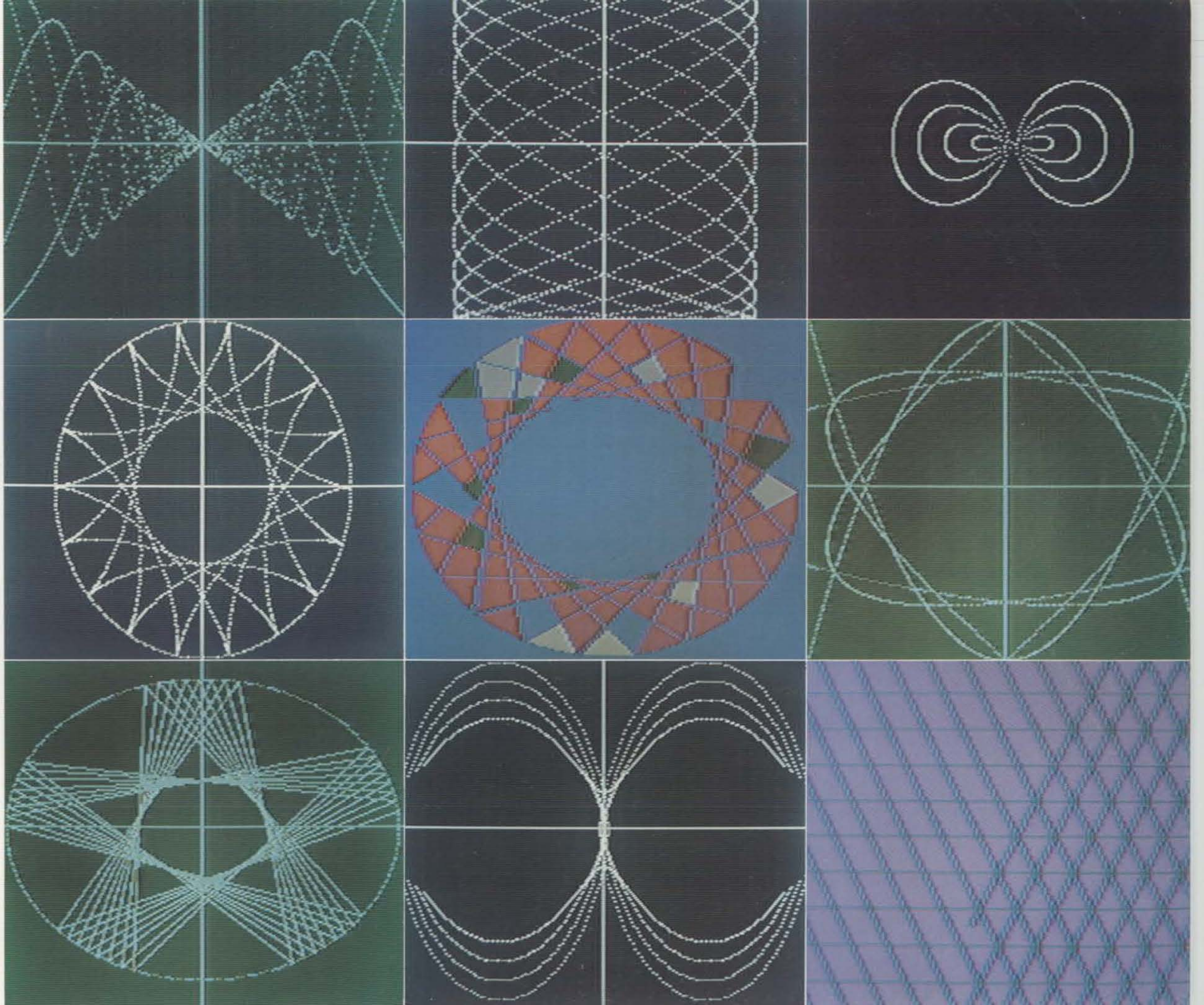






大きな波がやってくる．小さな波がとおりすぎる．
重なりそうで重ならないのは，いったい誰のせい？





第10章　きれいな絵を描く



さて，今までプログラムについて，いろいろと学んできたわけですが，それはさまざまな計算を繰り返し実行したり，その結果を表示したりするということでした．“計算機”なのですから，こういったことができるのは当然です．それでは，きみのPC-6001はそういった計算しかできない，つまり電卓の親分みたいな機械なのでしょうか．

もちろん，そうではありません．PC-6001は，画面にさまざまな絵やグラフを描くことができるのです．たとえば，カラーページの写真を見てください．きみにもこういった絵やグラフが，さらにお望みならもっと複雑な絵も描けるようになるのです．

それでは，順を追って，絵を描くN60-BASICの命令を学んでいきましょう．最後まで理解し終えたとき，きみにもマイコン・グラフィックスの世界が開けているのです．

10－1　PCの4つのモード

さて，いよいよグラフィック命令の登場ですが，この命令によって，今まで文字や数字，その他の記号しか表示されていなかった画面に，今度は突然絵が出てくるのでしょうか？

いえいえ，画面は，もともと文字しか書けないものであって，これから出てくるような絵を描くためには，そのための画面をPCに用意させる必要があるのです．そこで，まずその説明からまいりましょう．

PC-6001には文字だけの表示をするための画面の状態や，絵を描くための画面の状態があります．この画面の状態のことを“モード”といいます．そして，モードは次のように4つもあるのです．



モード1



文字を表示するためのモードです．これを“テキストモード”といいます．PCのスイッチを入れたときは，このモードになっているので，今までずっとこの画面を使っていたわけです．

モード2



文字をモード1と同じように表示できるほか，グラフィックスで，点や線も描けるモードです．このモードを“セミグラフィックモード”ともいいます．ただし，1つの点の大きさが大きく，1画面に横64個，縦48個の点しか描けません．

ですから，後で見るように，実際には点というより，四角といったほうが正確でしょう．また，文字と点を重ねて描くことはできません．どういうことかというと，文字の上に点を描こうとすると，文字は消えて点だけになってしまい，また点の上に文字を書こうとすれば，点は消えて文字だけになってしまうのです．

もちろん，画面上の異なった部分で，ここに文字，こっちは点や線のようにすることはできます．たとえば，棒グラフなどは簡単に作れます．

モード3

このモードでは，1画面に描ける点の数は，横128個，縦192個です．これは，相当細かいので，ゲームに使うとかなりおもしろいものができます．



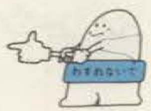
このモードは，“ローレゾリューション・グラフィックモード”とか，“ローレゾモード”，つまり，日本語でいうなら“低分解能グラフィックモード”といいます．

そして，文字を表示するときには，モード1やモード2と違って，文字を点に分解して1点ごとに画面に表示していきます（これは瞬時に行われるため，目で見てもわかりません）ので，文字の上に点を描いたり，文字を重ね書きしたりすることができます．

ただし，1文字を書くために横8，縦12の点が必要なため，1画面に書ける文字数は横16字，縦16行になってしまって，今までの文字に比べて横に2倍に伸びたように見えます．

モード4

このモードでは，1画面に横256個，縦192個の点が描けます．つまり，モード3と比べると，縦がそのままで横がちょうど2倍の細かさになっているのです．そして，“ハイレゾリューション・グラフィックモード”，略して“ハイレゾモード”，日本語では“高分解能グラフィックモード”と呼ばれています．



文字は，モード3と同じように分解して書かれますが，今度は横がモード3の倍ありますから，テキストモードと同様に1画面に横32字，縦16行の文字を書くことができます．

こうしてみると，なんだかモード3よりいいことだらけで，モード3はいったいなんのためにあるのか，と思うかもしれませんね．ところが，モード3では4色のカラー表示ができるのに，このモード4では1色しか使えないのです．ですから，目的に応じて，モード3とモード4を使いわける必要があるのです．

10−2　各モードの点をみてみよう

さて，前節でPCには4つのモードがあることはわかりましたね．しかし，話だけではわかりません．とにかく，やってみることが1番です．

そこで登場するのが，“SCREEN”命令です．この命令は，

```
SCREEN M,A,V
```



という形で使います．ここで，Mは1から4までの数で，モードを表すものです．もちろん，変数でもかまいませんよ．カンマで区切られたAとVも重要なものですが，説明は後回しにしますので，とりあえずは示されているとおりにキーインしておいてください．

まず，モード2にしてみましょう．そのためには，次のように入力します．

```
SCREEN 2,1,1
```



どうです．今まで緑色だったバックが黒になりましたね．これがモード2の画面なのです．何か適当に文字を打ちこんでみてください．

テキストモードとあまり変わらないのがわかるでしょう．これだけではつまらないので，点を打っ

てみましょう．説明は後でしますが，




```
PSET (0,0)
```

と打ちこんでみてください．すると“O”のあった画面の左上に，四角が出ましたね．これが，モード2の“点”です．実は，今の命令は画面の左上に点を描く命令なのです．

前に述べたように，点のところに書かれていた文字は消えてしまっていますね．そしてこの点は，1文字分のスペースに横2つ，縦3つの計6つが入る大きさになっています．ですから，あるところに点を打つと，1文字分のスペースが点を表示するために切り換わるので，その場所のまわりの6つ分が点を打てるようになって，文字は消えてしまうのです．

ではモード2は終わりにして，モード1に戻ることにしましょう．それには，

```
SCREEN 1,1,1
```

とすればいいですね．元の画面になったでしょう．

さて，次はモード3に挑みましょう．それには，

```
SCREEN 3,2,2
```

とします．ところが，一瞬画面がパッと変わるけれど，すぐ元に戻ってしまいますので何もやっていないようではありませんか．この命令はうそだったのでしょうか．

実は，命令はちゃんと実行されたのです．ところが，PCは入力待ちはモード1かモード2に戻るというしくみになっているのです．入力待ちというのは，画面にカーソルが表示されているときですね．現に画面ではカーソルが点滅しています．

それでは，モード3の画面を見るにはどうしたらよいのでしょう．そのためには，実行を終わらせなければよいのです．つまり，プログラムにして無限ループにしてしまうのです．次のプログラムを実行してみてください．

```
100 SCREEN 3,2,2
300 GOTO 300
```

GOTO文で無限ループに入ってしまうので，テキストモードに戻ることはなく，モード3の画面を表示し続けるわけです．この状態から脱出するには，STOPキーを押します．でも，これだけでは，何も表示していないのでおもしろくないですね．そこで，次のように追加してください．

```
100 SCREEN 3,2,2
110 PRINT "PC-6001"
300 GOTO 300
```

今度は，画面に文字が表示されましたね．しかも，今までの文字に比べて横に長い文字ですね．これが前に述べた，点の組み合わせで書いた文字なのです．今度は，点も描かせてみましょう．

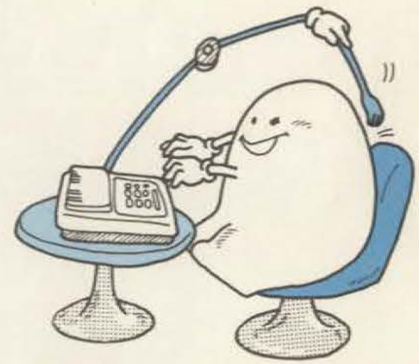
```
100 SCREEN 3,2,2:CLS
110 PRINT "PC-6001"
120 PSET (0,0)
300 GOTO 300
```

100行も変わっているのに注意してください．これはモード3の画面をクリアするためのものです．さっきのプログラムで，文字を書いてしまったので，それを消すために入れてあります．

では，実行してみてください．画面の左上に点を描きましたね．今度は，モード2のときと違い，本当に“点”らしく見えるでしょう．

さて，今度は文字の重ね書きをやってみましょう．次のプログラムを実行してみてください．

```
100 SCREEN 3,2,2:CLS
110 PRINT "OOOOO"
120 LOCATE 0,0
130 PRINT "*****"
300 GOTO 300
```



ここで，120行の命令は第2部第1章で出てくるものですが，カーソルの位置を左上に戻すものです．このため，“OOOOO”と“*****”は同じところに書かれることになり，“O”の中に“*”が入ったものが画面に現れるわけです．

そして，最後のモード4については，今までのプログラムの1部（1文字）を変えればいいのです．

たとえば，最初のプログラムなら，

```
100 SCREEN 4,2,2
110 PRINT "PC-6001"
120 PSET (0,0)
300 GOTO 300
```

となります．実行してみてください．左上すみだけ，他のすみと違っていますね．これがモード4の“点”なのです．

10－3　ページの切り換え

さて，この辺でまた命令の説明に戻りましょう．まだ，SCREEN命令の後2つの数が何か，という話が残っていました．



その前に，1つ覚えてほしい言葉があります．SCREEN命令の後の3つの数のような，命令に付けるものを，“パラメータ”といいます．そして，その1番目を“第1パラメータ”，2番目を“第2パラメータ”のようにいいます．覚えておいてくださいね．

それでは，SCREEN命令の第2，第3のパラメータの話に戻ります．これは，“ページ”というものを意味しています．PCは，ノートや本のページのように，あっちに何か書いて，こっちを見て，というようにいくつものページを扱い，そのうちの1ページを画面に表示することができるようになっているのです．

では，きみのPCは何ページあるのか，ということですが，それはスイッチを入れたときに，

```
How Many Pages?
```

で何と答えるかによるのです．ここで，単にRETURNキーを押したときは，最大限のページ数が用意されることになっています．そして，PC-6001本体だけのときは2ページ，ROM・RAMカートリッジ（PC-6006）を差しこんでいるときは4ページです．

さて，そこで第2パラメータは，文字や点を書きこむ命令があったとき，それを実際に書きこむページを示しています．これを，“アクティブページ”といいます．

そして，第3パラメータは，実際に画面に表示するページを示します．つまり，きみの目で見ているページで，これを“ビジュアルページ”といいます．



ですから，前にやった，

```
SCREEN 2,1,1
```

というのは，モードは2，書きこみを行うページと画面に表示するページは1ページ目にする，という意味だったのです．

ただし，1ページ目はモード1とモード2でしか使えず，モード3と4は2ページ目以降でしか使えません．もちろん，2ページ目以降では，モード1や2でも使えます．特別なのは，1ページ目だけです．ですから，前節では，

```
SCREEN 3,2,2
```

などとしたわけで，これを，

```
SCREEN 3,1,1
```

とすると，

```
?FC Error
```

になってしまいます．試しにやってみてください．これに対して，

```
SCREEN 2,2,2
```

は，何も問題がないわけですから，ちゃんと画面の左下に2という数字（2ページ目の“2”）が出ますね．ページの切り換えは，SCREEN命令以外にも，キーボードのモードチェンジキーでも行えます．



このように，N60-BASICではアクティブページとビジュアルページが別々に指定できるため，今見えているのと別の画面に文字や絵を書くことができます．たとえば，簡単な例として次のプログ

ラムを実行してみてください.

```
100 SCREEN 3,2,1
110 FOR I=1 TO 15:PRINT "わっ! おど
ろいた でしょ !":NEXT I
120 FOR I=1 TO 1000:NEXT I
130 SCREEN 3,2,2
300 GOTO 300
```

これは，100行でアクティブページをモード3の2ページ目にして，画面には1ページを表示しているようにします．次に文字を書きますが，これは見えないほうの2ページ目に書きます．そして，少し間をおいてから2ページ目をビジュアルページにしているので，画面にいきなり文字が登場するわけです．

なお，SCREEN 命令の3つのパラメータは一部を省略することができ，そのとき省略されたものは最後に指定したものとして扱われます．ですから，今のプログラムの

```
130 SCREEN 3,2,2
```

は，前の行の第1・第2パラメータをディフォルトとして

```
130 SCREEN ,,2
```

としてもよいのです．

また，前にも出てきた CLS 命令ですが，これは，アクティブページをクリアします．ですから，アクティブページとビジュアルページが違う場合には注意してください．たとえば，

```
100 SCREEN 3,2,1
110 FOR I=1 TO 15:PRINT "わっ! おど
ろいた でしょ !":NEXT I
120 FOR I=1 TO 1000:NEXT I
130 SCREEN ,,2
140 FOR I=1 TO 1000:NEXT I
150 SCREEN ,,1
160 CLS
170 FOR I=1 TO 1000:NEXT I
180 SCREEN ,,2
300 GOTO 300
```


プログラム続く



このプログラムでは，間をおいてビジュアルページを切り換えていますが，CLS 命令を実行する時点では，アクティブページである見えていないほうの2ページ目がクリアされるわけです．

では，画面切り換えの変わった例を紹介しましょう．次のプログラムです．やってみてください．




```
100 SCREEN 1,1,1:CLS
110 PRINT "N C P - 0 1"
120 SCREEN 1,2,2:CLS
130 PRINT " E   C 6 0 "
140 SCREEN ,1,1
150 SCREEN ,2,2
160 GOTO 140
```

10－4　色，いろいろ

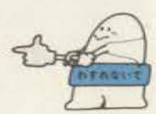
さて，今まで主に SCREEN 命令について試してきましたが，もちろんこれだけでは絵は描けません．そこで，点や線を描くことを説明しなくてはなりませんが，実はまだその前に説明しなければならないことがあります．この PC-6001 は，カラーグラフィックができるのです．ところが今まではいつも同じ色で描いてきました．ですから，まず色を変える命令について説明することにしましょう．

色を変える命令は，英語でそのまま，COLOR です．この命令の使い方は，




```
COLOR F,B,C
```

のようにします．



Fは，文字や点を書くときの色コードで“フォアグランドカラー”といいます．次のBは，背景の色コードで“バックグランドカラー”です．最後のCは色の組み合わせの選択コードで“カラーセット”と呼びます．

このコードの意味は，モードによって違いますから順にそれぞれの場合についてやってみましょう．なお，F·B·C の値は，全て整数なのですが，小数を使った場合は自動的に切り捨てられます．また，正しい範囲を超えた場合は，F·B に関しては，大きすぎるときは正しい値の最大に，小さす

ぎる場合は正しい値の最小になり，Cについてはエラーとなります．たとえば，モード3で，

```
COLOR 0,5,1
```

とすれば，

```
COLOR 1,4,1
```

と同じことになります．ただし，マイナスは許されません．つまり，今のを，

```
COLOR -1,5,1
```

としたのでは，

```
?FC Error
```

になってしまうのです．

それでは，コードの説明です．

モード1

Fは文字の色を表し，Bは何も意味がありません．また，Cは1と2のいずれかに応じて緑とオレンジが入れ わります．



モード1の色コード

F＼C	1	2
1	緑(グリーン)	オレンジ
2	緑(グリーン)のリバース	オレンジのリバース
3	オレンジ	緑(グリーン)
4	オレンジのリバース	緑(グリーン)のリバース

この表からもわかるように，Fは1から4までです．なお，色は必ずしもこのとおりにはなっていないかもしれませんが，それはテレビの色調整が違っているからです．PC-6001の取扱説明書をみて調整してください．

モード2

文字の色については，モード1のときと同じで，点については次のようになっています．



モード2の色コード

F,B ＼ C	1	2
0	黒（ブラック）	黒（ブラック）
1	緑（グリーン）	白（ホワイト）
2	黄（イエロー）	シアン
3	青（ブルー）	マゼンタ
4	赤（レッド）	オレンジ
5	白（ホワイト）	緑（グリーン）
6	シアン	黄（イエロー）
7	マゼンタ	青（ブルー）
8	オレンジ	赤（レッド）

ですから，モード2で，

```
COLOR 3,0,1
```

としたときは，文字はオレンジリバースとなり，点は青になるのです．また，Cを変えると，黒以外の8色が，4色ずつのブロックで入れ替わるわけです．今度は，F·Bの範囲は，0から8までになっ

ています．

モード3

モード2とは違い，文字と点の色は同じです．



モード3の色コード

F.B＼C	1	2
1	緑（グリーン）	白（ホワイト）
2	黄（イエロー）	シアン
3	青（ブルー）	マゼンタ
4	赤（レッド）	オレンジ

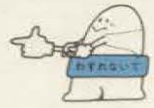
F·Bは，1から4までとなっています．たとえば，次のプログラムでは青地に赤で書かれるわけです．

```
100 SCREEN 3,2,2
110 COLOR 4,3,1
120 CLS
130 PRINT "NEC PC-6001"
300 GOTO 300
```

なお，FとBは別々に指定できるので，同じ色にしてしまうと見えなくなってしまいます．

モード4

モード4は最も細かい点が描けますが，色は次の表に示されるように，白と黒，または緑と黒の組み合わせだけです．



モード4の色コード

F,B ＼ C	1	2
0	黒(ブラック)	黒(ブラック)
1	緑(グリーン)	白(ホワイト)

このように，モード4ではF・Bは0か1だけです．たとえば，黒地に緑で書くならば，

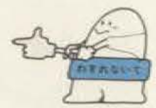
```
COLOR 1,0,1
```

となるわけです．

ここで，バックグランドカラーの役割について説明しておきましょう．まず，これが意味を持つのはモード2・3・4のときです．そして，CLS 命令を実行したとき，画面がこの色で塗りつぶされるわけです．だから，COLOR 命令を実行しただけでは，背景の色は変わらないので注意してください．

では，SCREEN 命令でモードを変えた後，いきなり点や文字を書くと何色になるのでしょう．実は，各モードで最後に実行した COLOR 命令が生きているのです．たとえば，モード3で白地にシアンとした後，モード4に切り換えて何かやって，その後またモード3に戻したとすると，前に設定した白地にシアンになるわけです．

また，スイッチを入れた直後は，モード3では



```
COLOR 2,1,1
```

が，モード4では，

```
COLOR 1,0,1
```

が，それぞれディフォルトになっています．

このように，プログラム実行前の状態によって変わってくるので，新しい SCREEN 命令の後に

は必ずCOLOR命令を付けるようにしてください.

COLOR命令は，以上のとおりですが，実際にいろいろ試しておくことにしましょう．まず，モード1の8とおりの色（実は2色とそのリバース）を見ましょう．

```
100 SCREEN 1,1,1
110 COLOR 1,0,1
120 CLS
130 FOR I=1 TO 2
140 FOR J=1 TO 4
150 COLOR J,0,I
160 PRINT "COLOR";J;", 0,";I;
170 PRINT " NEC PC-6001"
180 NEXT J,I
```

次はモード2です.

```
100 SCREEN 2,1,1
110 COLOR 1,0,1
120 CLS
130 FOR I=0 TO 8:COLOR I
140 PRINT "COLOR";I;",0,1"
150 NEXT I
160 FOR I=0 TO 8
170 PSET (110,I*12+5),I
180 NEXT I
```

また，Cを切り換えたものについては，次のプログラムです．

```
100 SCREEN 2,1,1
110 COLOR 1,0,2
120 CLS
130 FOR I=0 TO 8:COLOR I
140 PRINT "COLOR";I;",0,2"
150 NEXT I
160 FOR I=0 TO 8
170 PSET (110,I*12+5),I
180 NEXT I
```

そして，モード3についてやってみましょう．

```
100 SCREEN 3,2,2
110 FOR I=1 TO 4
120 COLOR I,I,1
130 CLS
140 FOR J=1 TO 4
150 COLOR J
160 PRINT "COLOR";J
170 PRINT " NEC PC-6001"
180 PRINT "  まいくろ・こんぴゅうたぁ"
190 FOR K=1 TO 100
200 NEXT K,J,I
210 GOTO 100
```

また，Cを変えると，

```
100 SCREEN 3,2,2
110 FOR I=1 TO 4
120 COLOR I,I,2
130 CLS
140 FOR J=1 TO 4
150 COLOR J
160 PRINT "COLOR";J
170 PRINT " NEC PC-6001"
180 PRINT "  まいくろ・こんぴゅうたぁ"
190 FOR K=1 TO 100
200 NEXT K,J,I
210 GOTO 100
```

最後は，モード4です．次の2つのプログラムを実行してみてください．

```
100 SCREEN 4,2,2
110 FOR I=0 TO 1
120 COLOR I,I,1
130 CLS
140 FOR J=0 TO 1
150 COLOR J
160 PRINT "      COLOR";J
170 PRINT "         NEC PC-6001"
180 PRINT "              まいくろ・こんぴ ゅ
うたぁ"
190 PRINT "                  Hi-Res
 Graphics"
200 FOR K=1 TO 100
210 NEXT K,J,I
220 GOTO 100
```

```
100 SCREEN 4,2,2
110 FOR I=0 TO 1
120 COLOR I,I,2
130 CLS
140 FOR J=0 TO 1
150 COLOR J
160 PRINT "      COLOR";J
170 PRINT "         NEC PC-6001"
180 PRINT "              まいくろ・こんぴ ゅ
うたぁ"
190 PRINT "                  Hi-Res
 Graphics"
200 FOR K=1 TO 100
210 NEXT K,J,I
220 GOTO 100
```

なお，カラーセットを変えると画面の色は全て切り換わってしまうため，たとえば，モード3では同一画面に赤とシアンを出すことはできません．次のプログラムで試してみてください．



```
100 SCREEN 3,2,2
110 COLOR 1,1,1
120 CLS
130 FOR I=1 TO 4
140 COLOR I
150 PRINT "  N E C"
160 PRINT "     P C -"
170 PRINT "      6 0 0 1"
180 NEXT I
190 COLOR ,,2
200 FOR I=1 TO 200:NEXT I
210 COLOR ,,1
220 FOR I=1 TO 200:NEXT I
230 GOTO 190
```

画面の色が，一斉に変化しますね．

また，COLOR命令もパラメータの省略ができます．今のプログラムで確認しておいてください．

10－5　点よ出ろ，点よ消えろ

いよいよ，絵を描くための命令とまいりましょう．さて，PC-6001に絵を描かせるためには，どうすればよいのでしょうか．“宇宙船の絵を描け”と命令すれば，すぐ描いてくれるのでしょうか．残念ながら，そのようなありがたい命令は，どんな立派なコンピュータにもありません．

画面のどこに，どんな図形をどのような大きさ・色で描けといったことを，細かく指定してやらなければ，いかにコンピュータといえどもお手上げなのです．

そこで，PCにはいったい何を指定してやればいいのでしょう．それには，PCのグラフィックスのしくみを思い出してみなければなりません．PCには4つのモードがあり，そのうちモード1以外の3つのモードでは，画面上に色付きの点を描くことができるのでしたね．

そして，モード3･4では，その点の組み合わせで文字を書いていたのでした．そうです．この“点の組み合わせで，ものを描く”ということこそ，全ての基本になるのです．PCの画面に描かれるはずの全ての図形は，有限個の点からなるわけですから，ある場所に点を打つことさえできれば，原理的には全ての図形が描けることになるのです．

そこで点を打つ命令，PSETが登場することになります．では，この命令を実行させるためには，どんな値を指定する必要があるでしょうか．それは画面上の位置と色ですね．これだけあれば，やるべきことは，はっきりしているので，PCが実行することができます．しかし，色は前にやったCOLOR命令があるので，後は位置をどうやって示すかが問題です．

画面は平面ですから，横，縦の2つの値で全ての位置を表現できますね．PCでは，横については，画面の左端が0，右端が255，縦については，上端が0，下端が191となっています．そして，必ず，カッコで囲んで横の方を先に書くことになっています．数学の座標平面（普通のグラフ）とは，上下逆になっているわけです．



ただし，小数の座標はないので小数点以下は切り捨てられてしまいます．また，範囲外の場合では，大きすぎるときはそれぞれの上限値になります．つまり，横は255，縦は191です．また，小さすぎるときは，負の値ですからエラーになります．

また，横が0から255までで，縦が0から191までになっているのは，最も細かいモード4が，横256

点，縦192点だからです．ただし，どちらも座標は0から始まっているので注意してください．
話よりやってみましょう．次のプログラムです．

```
100 SCREEN 4,2,2
110 COLOR 0,1,2:CLS
120 PSET (0,0):PSET (255,0)
130 PSET (0,191):PSET (255,191)
140 X=128:Y=96:I=1:J=-1
150 PSET (X,Y)
160 X=X+I:Y=Y+J
170 IF X=0 THEN I=1
180 IF Y=0 THEN J=1
190 IF X=255 THEN I=-1
200 IF Y=191 THEN J=-1
210 GOTO 150
```

実行する前にどうなるか考えてみてください．4すみに点を打って，次に中央からななめ上にどんどん点を打っていき，端までいったらはね返ったようになりましたか．そうならなかったら打ち間違いですから，プログラムを見直してください．

このように，点を打つのは，



PSET (X,Y)

で，横の位置，縦の位置の2つの値で座標を指定してやります．

さて，モード4では画面上の点の数が，横256，縦192ですから，左上すみの点が(0，0)，その1つ右が (1，0)，そのまた1つ右が (2，0)，……，となって，右上すみが (255，0) となったわけです．では，モード3や2ではどうなるのでしょうか．同じように，左上すみが(0，0)，その右が(1，0) で，……とやっていくと右上すみが，(　，0) や (63，0) になってしまい，その先がはみ出してしまいます．

それでは，どうなっているのかというと，まずモード3の場合は，(0，0) と (1，0) は同じ左上すみの点を示すのです．そして，その右の点は (2，0) または (3，0) で……と続いて，一番右上

の点は（254, 0）あるいは（255, 0）というふうに，1つの点に2つの座標があるわけです．

さらに，モード2では点が横64，縦48ですから，左上すみの点は（0, 0），（1, 0），（2, 0），（3, 0），（0, 1），（1, 1），（2, 1），（3, 1），（0, 2），（1, 2），（2, 2），（3, 2），（0, 3），（1, 3），（2, 3），（3, 3）の16の座標のどれでもよいというわけです．



なんだかややこしいですね．モードに応じて座標の値の範囲を変えて，1つの点に1つの座標，0の隣は1，1の隣は2というようにしたほうがいい，なんて思うかもしれません．

確かにそのほうが自然ですが，次のプログラムを実行すれば，そのわけがわかりますよ．

```
100 SCREEN 3,2,2
110 COLOR 1,2,2:CLS
120 PSET (0,0):PSET (255,0)
130 PSET (0,191):PSET (255,191)
140 X=128:Y=96:I=1:J=-1
150 PSET (X,Y)
160 X=X+I:Y=Y+J
170 IF X=0 THEN I=1
180 IF Y=0 THEN J=1
190 IF X=255 THEN I=-1
200 IF Y=191 THEN J=-1
210 GOTO 150
```

```
100 SCREEN 2,2,2
110 COLOR 1,0,2:CLS
120 PSET (0,0):PSET (255,0)
130 PSET (0,191):PSET (255,191)
140 X=128:Y=96:I=1:J=-1
150 PSET (X,Y)
160 X=X+I:Y=Y+J
170 IF X=0 THEN I=1
180 IF Y=0 THEN J=1
190 IF X=255 THEN I=-1
200 IF Y=191 THEN J=-1
210 GOTO 150
```

さきほどのモード4で試したプログラムを，モード3，モード2にそれぞれ書き変えたものですが，よく見るとたった3文字しか変えていないのがわかりますね．しかも，実行してみるとモード4のときの画面と感じが同じことがわかりますね．

このように，簡単にモードを変更できたのは，座標の値の範囲が，同じだったからなのです．もし，そうでなかったら，全面的に変更しなければならないことはわかりますね．このために，1つの点に座標が何とおりもあるという，一見不思議に思えるようなしくみになっているのです．

さて，今度はいろいろな色で点を描くことにしましょう．次のプログラムを実行してみてください．

```
100 SCREEN 3,2,2
110 COLOR 2,1,2:CLS
120 PSET (5,5)
130 COLOR 3
140 PSET (20,20)
150 COLOR 4
160 PSET (50,50)
300 GOTO 300
```

色を3つ使って点を描きましたね．モード3には4色ありますが，背景と同じ色で点を打っても無駄ですよ．

このように，色を次々と変えるにはCOLOR命令では不便ですね．そういうときは，次のようにします．

```
100 SCREEN 3,2,2
110 COLOR 2,1,2:CLS
120 PSET (5,5)
140 PSET (20,20),3
160 PSET (50,50),4
300 GOTO 300
```

このように，PSETの座標の後にカンマ(,)と色コードを付ければ，その色で点を描いてくれるのです（ちょっと点が小さすぎて見にくいかもしれませんが）．

さて，点を描くだけではいつか画面いっぱいに点を描いて終わってしまいます．描いた点を消す命令も必要ですね．特に，ゲームやコンピュータ・アニメーションでは，ものが動いていくようすを表現するために，一旦描いた点を消さなければなりません．このための命令が，PRESETなのです．使い方は，PSET命令と全く同じです．



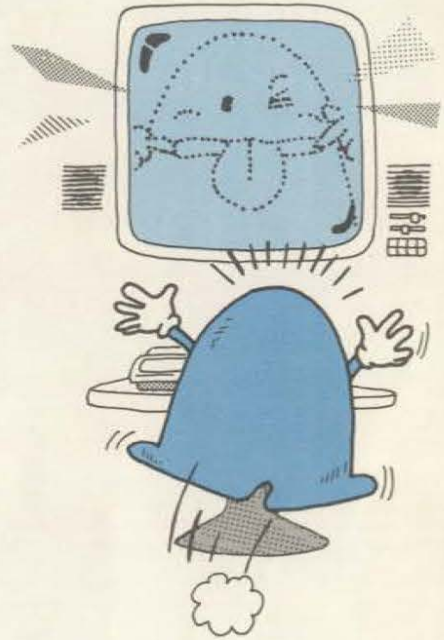
では，さっそくこれを使って，点を動かしてみましょう．

```
100 SCREEN 4,2,2
110 COLOR 0,1,2:CLS
120 X=128:Y=96:I=1:J=-1
130 PSET (X,Y):PRESET (X,Y)
```

プログラム続く



```
140 X=X+I:Y=Y+J
150 IF X=0 THEN I=1
160 IF X=255 THEN I=-1
170 IF Y=0 THEN J=1
180 IF Y=191 THEN J=-1
300 GOTO 130
```



どこかで見たような点の動きですね．でも，点がチカチカ点滅していて，あまり“動いている”という感じがしないですね．では，次のはどうですか．

```
100 SCREEN 4,2,2
110 COLOR 0,1,2:CLS
120 X=128:Y=96:I=1:J=-1
125 X0=127:Y0=97
130 PSET (X,Y):PRESET (X0,Y0)
140 X0=X:Y0=Y:X=X+I:Y=Y+J
150 IF X=0 THEN I=1
160 IF X=255 THEN I=-1
170 IF Y=0 THEN J=1
180 IF Y=191 THEN J=-1
300 GOTO 130
```

今度は，ちゃんと動いているように見えますね．さっきのと違うところは，点をいつ消すかということです．つまり，今度のは次の点を描いてから，前の点を消しているのです．

PRESET命令のしくみについて，さらに調べるために次の2つを実行してみてください．

```
100 SCREEN 3,2,2
110 COLOR 1,2,2:CLS
120 X=128:Y=96:I=1:J=-1
125 X0=127:Y0=97
130 PSET (X,Y):PRESET (X0,Y0)
140 X0=X:Y0=Y:X=X+I:Y=Y+J
150 IF X=0 THEN I=1
160 IF X=255 THEN I=-1
170 IF Y=0 THEN J=1
180 IF Y=191 THEN J=-1
300 GOTO 130
```

```
100 SCREEN 3,2,2
110 COLOR 1,3,2:CLS
120 X=128:Y=96:I=1:J=-1
125 X0=127:Y0=97
130 PSET (X,Y):PRESET (X0,Y0)
140 X0=X:Y0=Y:X=X+I:Y=Y+J
150 IF X=0 THEN I=1
160 IF X=255 THEN I=-1
170 IF Y=0 THEN J=1
180 IF Y=191 THEN J=-1
300 GOTO 130
```

これらは，さっきのプログラムをモード3にしたもので違いは，バックグランドカラーの指定だけです．どちらも同じように点が動いていきますが，よく考えてみると一方はシアンで消して，もう一方はマゼンタで消していることになります．

ということは，PRESET 命令は背景色で点を描くものと考えられますね．では，次のプログラムで確かめてみましょう．

```
100 SCREEN 3,2,2
110 COLOR 1,3,2:CLS:COLOR ,2
120 X=128:Y=96:I=1:J=-1
125 X0=127:Y0=97
130 PSET (X,Y):PRESET (X0,Y0)
140 X0=X:Y0=Y:X=X+I:Y=Y+J
150 IF X=0 THEN I=1
160 IF X=255 THEN I=-1
170 IF Y=0 THEN J=1
180 IF Y=191 THEN J=-1
300 GOTO 130
```

このプログラムでは，CLS 命令の前後で背景色を変更してしまっています．実行させると，まんまとPRESET 命令をだましてやったことがわかります．つまり，PRESET 命令は点を消すつもりで，シアンで点を描いてしまうのです．

つまり，Bをその時点での背景色とすれば，




```
PRESET (X,Y)
```

は，

```
PSET (X,Y),B
```

と同じということになります．確かめておきましょう．

```
100 SCREEN 3,2,2
110 COLOR 1,2,2:CLS
120 X=128:Y=96:I=1:J=-1
125 X0=127:Y0=97
130 PSET (X,Y):PSET (X0,Y0),2
140 X0=X:Y0=Y:X=X+I:Y=Y+J
150 IF X=0 THEN I=1
160 IF X=255 THEN I=-1
170 IF Y=0 THEN J=1
180 IF Y=191 THEN J=-1
300 GOTO 130
```

うまくいきましたね.

ところで, PRESET 命令にも, PSET 命令と同じく, カンマと色コードで消す色が指定できます.

ちょっと待ってくださいよ. その色で“消す”ということは, 実はその色で“描く”ということではありませんか. 試してみましょう.

```
100 SCREEN 3,2,2
110 COLOR 1,2,2:CLS
120 X=128:Y=96:I=1:J=-1
125 X0=127:Y0=97
130 PRESET (X,Y),1:PRESET (X0,Y0
),2
140 X0=X:Y0=Y:X=X+I:Y=Y+J
150 IF X=0 THEN I=1
160 IF X=255 THEN I=-1
170 IF Y=0 THEN J=1
180 IF Y=191 THEN J=-1
300 GOTO 130
```

このプログラムでは, 点を描くのも消すのも PRESET 命令でやっています.

さて, 今までのプログラムは, モード3や4で書いてきました. しかし, モード2では少し注意が必要です. ちょっと次のプログラムを実行してみてください.

```
100 SCREEN 2,2,2
110 COLOR 3,0,1:CLS
120 FOR X=0 TO 252 STEP 4
130 PSET (X,96):NEXT X
140 FOR Y=0 TO 188 STEP 4
150 PSET (128,Y):NEXT Y
160 PSET (128,96),4
```

これは，左から右へ，上から下と青で線を描いていって，最後にその交点を赤くするもののはずです．しかし，どうでしょう．中央の交点だけでなく，その右1つ，下2つの計4つの点が赤くなってしまいました．

これはなぜかというと，モード2では1の文字分のスペースに縦3つ，横2つの計6つの点が入るわけですが，実はこの6つの点は1組になっていて，全部同じ色にしかならないのです．そして，この例では赤くなった4つの点がその1組だったわけです．

10－6　点はあるか，ないか，それが問題だ！

もう，点を描いたり，消したりはきみの思いのままです．しかし，ゲームなどではあるところに点があるか，ないかを知りたいことがあります．



N60-BASICには，そのためにPOINTという関数が用意されています．この関数は，座標（X, Y）にある点の色コードを教えてくれるものです．とにかく，試してみましょう．

```
100 SCREEN 2,2,2
110 COLOR 1,0,1:CLS
120 FOR X=1 TO 8
130 PSET ((X-1)*32,32),X
140 NEXT X
150 FOR X=0 TO 252 STEP 4
160 C=POINT (X,32)
170 IF C=0 THEN 200
180 C=C+1:IF C=9 THEN C=1
190 PSET (X,32),C
200 NEXT X
210 GOTO 150
```

これは，画面に8色点を描いた後に，その色を調べて色を変えていくものです．ただし，モード2では文字の書かれている場合は，POINT関数の値が0になってしまいます．そこで，0だった場合は次の点を調べています．

もう少し複雑な例を見てみましょう．



```
100 SCREEN 3,2,2
110 COLOR 1,2,2:CLS
120 LINE (50,50)-(200,150),4,B
130 LINE (60,60)-(100,80),4,B
140 LINE (140,80)-(190,130),4,B
150 LINE (70,110)-(120,140),3,B
160 X=128:Y=96:I=2:J=1:C0=2
170 PSET (X,Y)
180 X0=X:Y0=Y:C1=C0:X=X+I:Y=Y+J
190 C=POINT (X,Y)
200 IF C=4 THEN IF POINT(X+I,Y)=
4 THEN J=-J:Y=Y+2*J:GOTO 230
210 IF C=4 THEN IF POINT(X,Y+J)=
4 THEN I=-I:X=X+2*I:GOTO 230
220 IF C=4 THEN I=-I:J=-J:X=X+2*
I:Y=Y+2*J
230 C0=POINT(X,Y)
240 PSET (X,Y):PSET (X0,Y0),C1
250 GOTO 180
```

始めのほうに，見慣れない命令が並んでいますが，これについては次の節でやります．実行してみればわかりますが，四角を描く命令なのです．そして，前と同じように点を動かしていくわけですが，POINT関数で点の行く先を調べて，それが4，つまりオレンジなら反射し，それ以外のときは，すどおりして元のままの色にしておくわけです．

少し複雑なプログラムなので説明しましょう．始めの画面設定の後にループに入ります．点の進行方行を先取りして，まず横にのびている壁を判定して，次に縦にのびている壁を判定します．そして，1番外の箱から飛び出さないように，コーナーに内側からあたったときの判定をしています．

重要なことは，先取りしたデータを使って点の通ったあとを元どおりにしておくところです．た

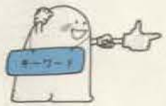
だし，このプログラムは，中のオレンジの四角のコーナーにある角度であたると反射せずに，中へ入ってしまいます．その理由は考えてみてください．

10－7　線と箱

点が描ければ，どんな図形でも描けるはずです．しかし，全ての図形を1つずつ点で描いていくのでは非常にめんどうです．

N60-BASICでは，そのために基本的な図形を描く命令が用意されています．それは，線分，中の空いた長方形，中の詰まった長方形の3つです．そして，これらは全て同じLINEという命令で行われるのです．

線分の場合，その両端の点を指定しなければなりません．そこで，(A, B) の点から，(C, D) の点まで線分を引くことを考えてみましょう．それには，

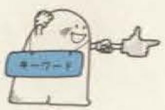



```
LINE (A,B)-(C,D)
```

とすればよいのです．ここで，真中の"－"は，キーボードの0の右にあるマイナス符号で，￥のかなシフトした"－"ではありません．

また，このLINE命令も，PSETやPRESETと同様に，色コードの指定をカンマに続けて追加することができます．もちろん，この指定がないときは，PSET同様にCOLOR命令のフォアグラウンドカラーで描かれるわけです．

始めの (A, B) を省いて，




```
LINE -(C,D)
```

と書くと，これはその直前にLINE命令，またはPSET命令で描いた点から，座標 (C, D) の点まで線分を引くことができます．では，実際にやってみましょう．

```
100 SCREEN 4,2,2
110 COLOR 1,0,1:CLS
120 X=128:Y=96:R=60
130 PSET (X+R,Y)
140 FOR T=2.55 TO 85 STEP 2.55
150 LINE -(X+R*COS(T),Y+R*SIN(T)
)
160 NEXT T
300 GOTO 300
```

これは，(128, 96)を中心として半径60の円周上に点を取って，LINE命令でつないでいくものです．では，次です．

```
100 DEF FNR(X)=RND(1)*X
110 SCREEN 3,2,2
120 COLOR 1,1,2:CLS
130 PSET (FNR(256),FNR(192)),FNR
(4)+1
140 LINE -(FNR(256),FNR(192)),FN
R(3)+2
150 GOTO 140
```

これは，乱数で点を選んで，その間を乱数による色の線分でつないでいくものです．

では，次は長方形のわくを描く命令です．これは今のLINE命令の後に“,B”を付け加えればいいのです．つまり，

```
LINE (A,B)-(C,D),1,B
```



とすれば，この2点を対角線とする，画面の4辺とそれぞれ平行な長方形を，色コード1で書くわけです．つまり，この命令は次の4命令を合わせたものと同じ機能を持っています．

```
LINE (A,B)-(C,B),1
LINE (A,D)-(C,D),1
LINE (A,B)-(A,D),1
LINE (C,B)-(C,D),1
```

また，色コードを指定しない場合は，



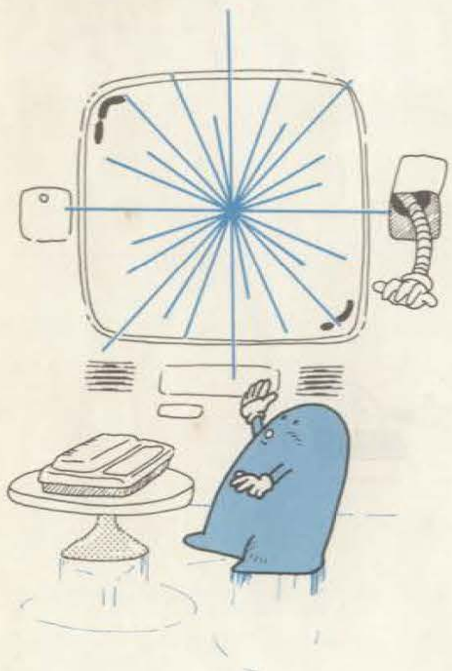
```
LINE (A,B)-(C,D),,B
```

のように書くことができます．この最後の“B”は“ボックス (box)”の意味です．なお，これを，

```
LINE (A,B)-(C,D),B
```

としてしまうと，Bという変数の値を色コードとして線分を描く命令になってしまいますので，間違えないようにしてください．

次に中の詰まった長方形は，今の“B”の代わりに“BF”とします．こちらは，“ボックスフィル (box fill)”の意味です．それでは，次のプログラムで，LINE命令の3とおりの使い方を，しっかり目で確かめておいてください．



```
100 SCREEN 3,2,2
110 COLOR 4,1,2:CLS
120 LINE (10,10)-(250,50)
130 LINE (10,60)-(250,100),,B
140 LINE (10,110)-(250,150),,BF
300 GOTO 300
```

また，次のプログラムは，3色でランダムに，中ぬきの，あるいは中の詰まった長方形を順に描いていくものです．

```
100 SCREEN 3,2,2
110 COLOR 1,1,2:CLS
120 FOR X=1 TO 15:FOR Y=1 TO 11
130 IF RND(1)<.5 THEN 150
140 LINE (X*16-6,Y*16-7)-(X*16+6
,Y*16+7),RND(1)*3+2,B:GOTO 160
150 LINE (X*16-6,Y*16-7)-(X*16+6
,Y*16+7),RND(1)*3+2,BF
160 NEXT Y,X:GOTO 120
```

そしてもう一つ，ランダムに，中ぬきの長方形を描いていくプログラムです．

```
100 SCREEN 3,2,2
110 COLOR 1,1,2:CLS
120 LINE (RND(1)*256,RND(1)*192)
-(RND(1)*256,RND(1)*192),RND(1)*
3+2,B
130 GOTO 120
```

最後に，これら3つのLINE命令がなくて，PSET命令しかなかったとしたら，どうなるかというプログラムをあげておきましょう．

```
100 SCREEN1,1,1:COLOR 1,0:CLS
110 PRINT "LINE (A,B)-(C,D),3 --
--------1"
120 PRINT "LINE (A,B)-(C,D),3,B
--------2"
130 PRINT "LINE (A,B)-(C,D),3,BF
 -------3"
140 INPUT "どれに しますか";X
150 INPUT "A,B,C,D=";A,B,C,D
160 SCREEN3,2,2:COLOR 3,1,2:CLS
170 IF X=1 THEN GOSUB 310
180 IF X=2 THEN GOSUB 410
190 IF X=3 THEN GOSUB 510
200 FOR I=1 TO 500:NEXT I:GOTO 1
00
300 REM LINE (A,B)-(C,D)
310 K=(ABS(A-C)+ABS(B-D)+ABS(ABS
(A-C)-ABS(B-D)))/2
320 IF K=0 THEN PSET (A,B):RETUR
N
330 FOR I=0 TO K
340 PSET (A+(C-A)/K*I,B+(D-B)/K*
I)
350 NEXT I:RETURN
400 REM LINE (A,B)-(C,D),,B
410 A0=A:A=C:GOSUB 300
420 B0=B:B=D:A=A0:GOSUB 300
430 B=B0:C0=C:C=A:GOSUB 300
440 D=B:C=C0:GOSUB 300
450 RETURN
500 REM LINE (A,B)-(C,D),,BF
510 X0=1:IF A>C THEN X0=-1
520 Y0=1:IF B>D THEN Y0=-1
530 FOR X=A TO C STEP X0
540 FOR Y=B TO D STEP Y0
550 PSET (X,Y)
560 NEXT Y,X:RETURN
```

プログラム続く



LINE命令の非常にありがたいことがわかるでしょう．

10－8　ぬり絵は得意？

さて，今度は円に挑んでみましょう．まず，円は三角関数で計算できるので，次のようなプログラムが考えられます．



```
100 SCREEN 4,2,2
110 COLOR 1,0,1:CLS
120 X=128:Y=96:R=50
130 FOR T=0 TO 3.142*2 STEP .02
140 PSET(R*COS(T)+X,R*SIN(T)+Y)
150 NEXT T
300 GOTO 300
```

しかし，やってみればわかるように，非常に遅くていらいらしてきます．そこで，次のようにしてみました．

```
100 SCREEN 4,2,2
110 COLOR 1,0,1:CLS
120 X=30:Y=0:A=1/32
130 FOR I=1 TO 200
140 X=X-A*Y:Y=A*X+Y
150 PSET (X+128,96-Y)
160 NEXT I
300 GOTO 300
```



どうです．前よりずっと速くなったでしょう．それでは今度は，円周だけではなく，中に詰まった円を描きたいと思ったらどうしたらよいでしょう．

まず，次のようなものが考えられます．

```
100 SCREEN 4,2,2
110 COLOR 1,0,1:CLS
115 FOR J=30 TO 0 STEP -1
120 X=J:Y=0:A=1/32
130 FOR I=1 TO 200
140 X=X-A*Y:Y=A*X+Y
150 PSET (X+128,96-Y)
160 NEXT I,J
300 GOTO 300
```

つまり，半径をだんだん小さくした円を描いていくのです．しかし，これでは非常にめんどうです．いちいちこんなことをしていたのでは，たまったものではありません．

そこで，登場する命令が，PAINTです．これは，ある領域をぬりつぶすための命令で，次のような形で使います．

```
PAINT (X,Y),C,B
```



そして，Bは境界色，Cはぬりつぶす色です．つまり，Bの色で囲まれた領域で，(X,Y)の点を内部に持つ部分を，Cの色でぬりつぶすわけです．

そこで，この命令を使えば，さっきのプログラムは次のような簡単な形になるわけです．

```
100 SCREEN 4,2,2
110 COLOR 1,0,1:CLS
120 X=30:Y=0:A=1/32
130 FOR I=1 TO 200
140 X=X-A*Y:Y=A*X+Y
150 PSET (X+128,96-Y)
160 NEXT I
170 PAINT (128,96),1,1
300 GOTO 300
```

断然，この方が速いですね．

今度は，モード3でやってみましょう．次のプログラムです．色が4色あるので，ちょっと前のより複雑なことができます．

```
100 SCREEN 3,2,2
110 COLOR 2,1,2:CLS
120 LINE (250,10)-(10,150),4
130 LINE (150,70)-(130,120)
140 LINE -(70,50)
150 LINE -(150,70)
160 PAINT (128,115),3,2
300 GOTO 300
```

オレンジの線が無視されて，シアンだけが境界と考えられていることがわかります．

さて，PAINT命令は内部をぬる命令でしたが，今のプログラムで，座標の値として3角形の外部の点を指定したらどうなるでしょう．やってみましょう．

```
160 PAINT (10,10),3,2
```

3角形の外部をぬりつぶしてしまいましたね．つまり，指定された点を元に考えて境界のところまでぬりつぶすのですから，こういう結果になるのは当然ですね．それでは，今度は境界上の点を指定してみましょう．

```
160 PAINT (70,50),3,2
```

どこもぬらなかったようです．始めから，そこに境界の色があれば，もうそれ以上ぬることはないわけです．

それでは，3角形を少し開いたものを考えてみましょう．次のプログラムです．

```
100 SCREEN 3,2,2
110 COLOR 2,1,2:CLS
120 LINE (250,10)-(10,150),4
130 LINE (150,70)-(130,120)
140 LINE -(70,50)
150 LINE -(150,68)
160 PAINT (128,96),3,2
300 GOTO 300
```

今度は，3角形の線の上を除いた全画面をぬりつぶしてしまいました．つまり，境界につきあたるまでとぬり続けているうちに，全画面をぬりつぶしてしまったのです．次のもその例です．

```
100 SCREEN 3,2,2
110 COLOR 2,1,2:CLS
120 LINE (150,70)-(130,120)
130 LINE -(70,50):LINE -(150,70)
140 LINE (250,10)-(10,150),4
150 PAINT (128,96),3,2
300 GOTO 300
```

1番目のプログラムの，順序を変えただけですが，今度は3角形の辺を除く全画面をぬりつぶしてしまいます．これはなぜかというと，3角形の辺と，後で引いた直線が交わった点では，後から描いた色であるオレンジになってしまうからです．そのため，境界色でみれば，穴が開いていることになってしまったからなのです．

なお，このPAINT命令については，境界色もぬりつぶす色のコードも，省略できません．

また，前節の中の詰まった長方形を描く命令は，わくだけ描いておいて，その後，このPAINT命令で中をぬりつぶす方法によっても描くことができます．それでは，比べてみましょう．

```
100 SCREEN 3,2,2
110 COLOR 2,1,2:CLS
120 LINE (20,20)-(240,70),,BF
130 LINE (20,100)-(240,150),,B
140 PAINT (22,101),2,2
300 GOTO 300
```

どうです．グラフィック命令を使いこなせるようになれば，複雑なゲームやきれいな絵が描けるようになります．わかるまで繰り返し読んで，そして何より実際にやってみてくださいね．

```
100 SCREEN 3,2,2
110 COLOR 1,2,2:CLS
120 FOR X=0 TO 250 STEP 25
130 LINE (X,0)-(X,190)
140 NEXT X
150 FOR Y=0 TO 190 STEP 19
160 LINE (0,Y)-(250,Y)
170 NEXT Y
180 LINE (252,0)-(252,191)
190 LINE (254,0)-(254,191)
200 LINE (0,191)-(254,191)
210 PAINT (INT(RND(1)*10)*25+10,
INT(RND(1)*10)*19+10),RND(1)*3+2
,1
220 GOTO 210
```



```
100 SCREEN 3,2,2
110 COLOR 1,2,2:CLS:PSET (0,0)
120 FOR I=1 TO 50:R=RND(1)
130 IF R<0.25 THEN X=0:Y=RND(1)*
192:GOTO 170
140 IF R<0.5 THEN X=255:Y=RND(1)
*192:GOTO 170
150 IF R<0.75 THEN Y=0:X=RND(1)*
256:GOTO 170
160 Y=191:X=RND(1)*256
170 LINE -(X,Y)
180 NEXT I
190 FOR I=1 TO 100
200 PAINT (RND(1)*256,RND(1)*192
),RND(1)*4+1,2
210 NEXT I
220 GOTO 120
```

```
100 SCREEN 4,2,2
110 COLOR 0,1,2:CLS
120 R=20:P=3.14159265
130 FOR I=1 TO 50
140 X=RND(1)*200+25:Y=RND(1)*150
+20
150 PSET (X+R,Y)
160 FOR J=1 TO 5
170 LINE -(X+R*COS(J*P*2/5),Y+R*
SIN(J*P*2/5))
180 NEXT J,I
190 PAINT (RND(1)*256,RND(1)*192
),RND(1)*2,RND(1)*2
200 GOTO 190
```



第11章　これがほんとのコンピュータ音楽



クリック音の出ている所はどこですか．そうです．PC-6001の上面，左側奥の正方形に細いスリットのある所ですね．この下に内蔵スピーカがあり，そこからこのクリック音が出ています．

そして，ボリューム・コントロールツマミは，PCの裏側のちょうどスピーカの後ろにありますね．何かキーを押しながら，クリック音の大きさを小さめに聞こえるように調整しておいてください．

このスピーカから何か音楽でも出てくると，いっそうプログラミングが楽しくなりそうです．また，ゲームの効果音として，ミサイルの爆発音なんかが出れば，ゲームはぐんとおもしろくなりますね．

11－1　簡単に音楽演奏するには

これはいちいち説明するより，実際にやってみるのが，最も近道な方法ですから，次のように入力してみてください．

```
PLAY "C"
```

どうです．“ド”の音が出ましたね．では，次に，

```
PLAY "C","E"
```

“ド”と“ミ”の和音です．さらに，

```
PLAY "C","E","G"
```

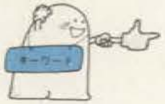
もちろん，“ドミソ”の和音です．



```
PLAY "C","E","G","B"
```

今度は，どうですか．思っていたのとは違う音を出して，しかもごていねいに，

```
?SN Error
```



のおまけ付きですね．もうわかりました．この PLAY という命令は音楽を演奏するためのもので，同時に3つの音を出すことができるのです．



さて，CEG がそれぞれドミソに対応していましたね．きみの良く知っている“ドレミファソラシ”は，正式には “CDEFGAB” で表し，これらを音の名前で “音名” といいます．それでは本当にそうなっているのか，確かめておきましょう．

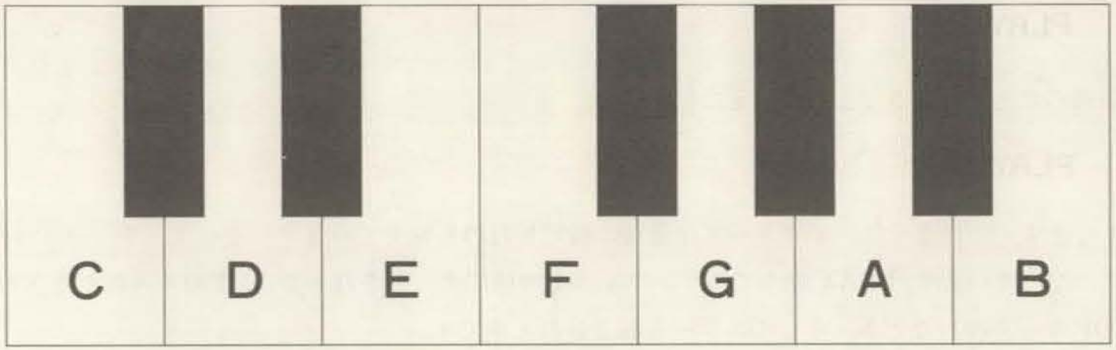



```
PLAY "cdefgab"
```

あっ！ 間違えて小文字で打ってしまいましたが，大丈夫のようです．大文字でも小文字でもかまわないのです．ちゃんとドレミ……が演奏できましたね．しかも，RETURN を押すと音が出ているのにカーソルが現れました．もう1度ドレミをやらせておいて，カーソルが現れたらすぐに，今打ちこんだ行に移動してから，RETURN を押してみてください．

2回ドレミが演奏できましたね．このPLAYという命令は，残りの音がある程度少なくなると次の処理を始めることができる特殊な命令なのです．良く意味がわからない人は，これはすごい命令なんだぐらいに思っていてください．別にわからなければ使えないというわけではないですから．

さて，"C"から"B"までの1オクターブは，音名では7つしかありませんが，実は12の音からなっていることは，何か楽器を操ったことのある人なら知っているでしょう．たとえば，ピアノの鍵盤を思い出してください．



1オクターブの中に白い鍵は7つで，それぞれ音名のCからBまでに割り当ててありますが，その間に黒い鍵が5つあります．この黒鍵の音はその両隣の中間の高さで，下（左）の音からは半音高く，上（右）の音からは半音低くなっています．

そして，半音上げる記号のシャープ（♯），半音下げる記号のフラット（♭）を使って，隣の音名で表されます．たとえば，CとDの間の黒鍵は，C♯またはD♭で表されます．これをPLAY命令では，"＋"と"－"の記号を使って表します．試しに次の2つを打ちこんでみてください．




```
PLAY "CC+DD+EFF+GG+AA+B"
PLAY "BB-AA-GG-FEE-DD-C"
```

これで，1オクターブの音の高さの関係はわかりました．しかし，ピアノではこの右にも左にも何オクターブ分もの鍵盤が用意されています．ちょっといい曲を演奏しようと思ったら，当然1オクターブの音で足りるわけはありません．



そこで，PLAY命令では，オクターブ(Octave)の“O”を使って，1オクターブ全体をオクターブ単位で指定することができます．何はともあれやってみましょう．

```
PLAY "O3CO4CO5CO6C"
```
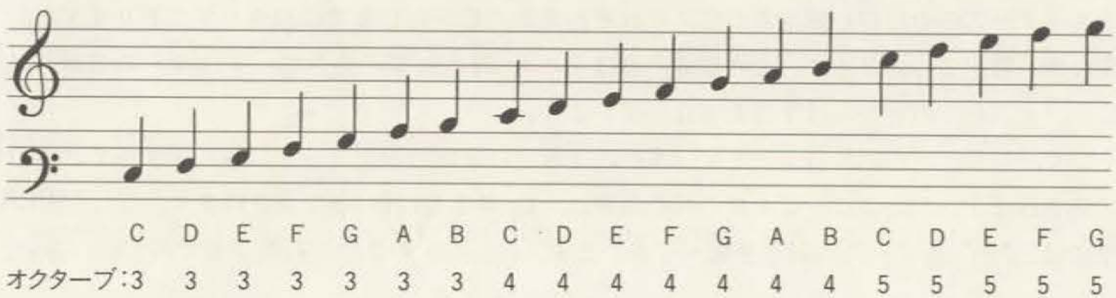
4つの音が出ましたが，すべての“C”の音です．しかし，高さが全然違いますね．それぞれ，1オクターブずつ高くなっているのです．ここで，ただ，“C”だけを演奏してみましょう．

```
PLAY "C"
```

最後の音と同じ高さでしたね．元に戻すには，

```
PLAY "O"
```

とします．このように，オクターブは最後に指定したのを覚えています．そして，ディフォルトは4になっています．PLAY命令ではディフォルトの値にセットされるのは，電源を入れたときと，STOPキーを押したとき，そしてエラーを起こしたときです．



PCが出せる1番低い音，そして1番高い音はなんでしょう．まず，1オクターブの中で1番低い音は“C”，1番高い音は“B”ですね．それでは，次のように打ってください．

```
PLAY "O3C"
```

この“3”を1ずつ小さくしていくと，

```
PLAY "O0C"
```

のところで，

```
?FC Error
```

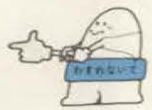


が出ましたね．1番低い音は“O1C”だというわけです．それでは，

```
PLAY "O6B"
```

と打っておいて，“6”のところを1ずつ増やしてみてください．こんどは，

```
PLAY "O9B"
```



で，エラーが出ました．つまり，1番高い音は“O8B”ですね．PCは全部で，8オクターブもの音域があるのです．普通のピアノは6オクターブくらいですから，それより低い音や高い音が出せるのです．きみはこれらの音がちゃんと聞こえましたか．

さて，音の高さはこれできみの思いのままです．それでは，何かの楽譜を持ってきて演奏させてみましょう．どうです．うまく演奏できましたか．

楽譜を見て思い出しました．まだ，音の長さを変えられませんでしたね．今の音は全部同じ長さで，2つ以上同じ音を並べるとその数だけつながって出ます．たとえば，

```
PLAY "CDDEEE"
```

このように，いくらでも長い音は出せますね．しかし，もっと短い音は出ないのでしょうか．だいじょうぶ，ちゃんと出ます．実は，今の音の長さは1音で，テンポが120のときの4分音符の長さなのです．そして，各音名の後に4が省略されています．すなわち，次の2つは同じです．

```
PLAY "CC+"
PLAY "C4C+4"
```

さあ，それなら，8分音符はこの“4”のところを“8”にすればよさそうです．やってみましょう．

```
PLAY "C8C+8"
```

どうです．さっきの半分の長さになりましたね．普通音符には次のようなものがあります．



全音符	2分音符	4分音符	8分音符	16分音符	32分音符	64分音符
1	2	4	8	16	32	64



PLAY命令では，音名の後ろに上の数字を付ければよいのです．もちろん，全音符が1番長く，64分音符が1番短いわけです．そして，それぞれ左隣の半分ずつの長さになっています．

まず，普通の楽符には出てきませんが，128分音符というのもあります．これはどうでしょうか．

```
PLAY "C128"
```

どうです．エラーが出ましたね．

さて，音符には他にも，付点音符や3連符などがでてきます．まず付点音符は，その音符よりさらに半分長いのですから，たとえば付点4分音符なら，4分音符に8分音符を加えた長さということになります．すなわち，

```
PLAY "C4C8"
```

で表すことができますが，これではちょっとめんどうです．付点というからには長さの後に点を付けて表せれば，ずっとわかりやすいですね．もしかしたらできるかもしれません．考えているより，とにかくやってみましょう．点は“ピリオド”でしょう．




```
PLAY "C4."
```

さきほどのと同じくらいの長さですね．本当にそうなのか試すには，2つの音を同時に並べて演

奏すればいいですね.

```
PLAY "C4.","E4E8"
```
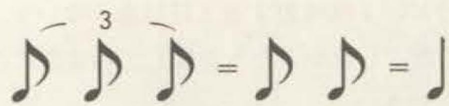
どうやら, まったく同じ長さのようですね. さらに点を付けていくとどうなるでしょう.

```
PLAY "C4..","E4E8"
```

どうやら, "ド" の方が長かったようです. どれくらい長いかは, "ミ" の音を少しずつ変えればわかります. やってみてください.
さて, 正解は次のようになりましたね.

```
PLAY "C4..","E4E8E16"
```

このように, 点を付けていくと本来の音の長さの半分, さらに半分というぐあいに伸びていきます. 3つ, 4つと点を付けたらどうなるかは, ぜひ実際にやっておいてください.
次に, 3連符は3個でその音符の本来の長さの2個分にあたりますから, たとえば8分音符の3連符では, その1つ1つの長さは8分音符2個分のさらに3分の1, すなわち, 12分音符の長さになるわけです.



たとえば, "ドミソ" の8分音符の3連符は, "C12E12G12" となります. 本当にそうなるのか確かめておきましょう.

```
PLAY "C12E12G12","C8E8","C"
```

さて, もう1度楽譜をながめてみましょう. 五線譜の中には音符に混ざって, 休符があります. これはPLAY命令ではどうしたらよいのでしょう. 心配しないでください. ちゃんとあります. 休符はレスト (Rest) の "R" で示します.



では次のように打ちこんでみてください.

```
PLAY "RC","CD"
```

どうです．２番目の音が重なりましたね．普通の楽譜には，次のような休符があります．



全休符	2分休符	4分休符	8分休符	16分休符	32分休符	64分休符
1	2	4	8	16	32	64

PLAY命令では，音符の長さと同じように，休符の長さも“R”の後ろに書けばよいのです．

```
PLAY "R8C","C8D"
```

２つの音の終わりが同じですから，これは８分休符を示すことがわかりますね．

音符の長さを指定しないときは，全部４分音符になりました．しかし，もし８分音符が多い曲を演奏しようとしたときは，これでは不便ですね．そこで，音符の長さを指定しないときの決まった長さを変更できれば，非常に便利だと思いませんか．



そこで，このPLAY命令には親切にもそのために“L”というのを用意してあります．これは音の長さ（Length）からきたもので，“　”の後ろにその曲で１番多く使う音符の長さを指定しておけばよいのです．これもオクターブ指定の“O”と同じように，１度変更すると以後変えないか，ディフォルトの４にセットされない限り，音名の後に長さを指定しなかったときは，この長さになります．実際にやってみましょう．次のようにして試してください．

```
PLAY "L4CDEL8CDE"
```

それでは，休符はどうでしょう．やってみましょう．

```
PLAY "L8RC","L8CD"
```

おや！　考えていたのとはちょっと違うようですね．これは，

```
PLAY "L8R4C","L8CD"
```

と同じです．つまり，休符は指定しないといつでも4分休符で，この“L”は音符にだけ関係するということがわかりますね．

さて，PLAY命令の機能がわかったところでいよいよ簡単に曲を演奏してみましょう．その前に，いちいち“PLAY……”と打ちこんでいたのでは，ちょっと長い曲になると疲れてしまいますし，後で修整するときにたいへんそうです．

“PLAY”の後には，カンマで区切られた3つまでの文字列が並んでいればいいわけですね．この文字列を文字変数に置き換えられないでしょうか．そこで，まず，

```
A$="CDE"
```

としておいて，

```
PLAY A$
```

とするというのは，どうでしょうか．それぞれやってみてください．

どうです．うまくいきましたね．ただし，必ず文字変数ですよ．たとえば，

```
PLAY A
```

では，

```
?TM Error
```



になってしまいます．これを使って曲を演奏するためのプログラムを作ってみましょう．次のようなものを考えてみました．




```
100 REM 'DATA' めいれい を つかって、
110 REM きょく の データ を 、200 ぎょう
120 REM の れい の ように いれてください。
130 READ A$
140 IF A$="stop" THEN END
150 READ B$,C$
160 PLAY A$,B$,C$
170 GOTO 130
200 DATA CRCRO3B,ERFRG,GRARF
210 DATA stop
```

使い方は，プログラムをみれば簡単にわかりますね．DATA 文で曲を入れればいいんです．最後に必ず，“STOP” を入れておくのを忘れないでください．

11－2　曲に変化をつけよう

前節を読んだきみなら，もうどんな曲でも PC で演奏することができますね．しかし，何かものたりないとは思いませんか．そうです．まだまだ音楽にはいろいろな要素があるのです．

PC の PLAY 命令にも，まだ説明していない機能があります．これを使えば，人を感動させるような名曲を奏でることもできるでしょう．

さて，音楽にはいろいろな速さのものがあります．曲の速さを変えるには，音符の長さをそれぞれ同じ割合で変えればいいわけです．しかし，すでにプログラムに入力した曲の音符の長さを，1つ1つ変えていたのではたいへんな手間がかかってしまいます．

普通の楽譜を見ると，五線譜の左上に “♩＝120” とか，“♩＝60” とかが書かれています．これがその曲の全体の速さ，つまり “テンポ” を示しているのです．たとえば，“♩＝120” と書かれていたら，1分間に4分音符を120個演奏する速度ということになります．



PLAY 命令では，テンポ（Tempo）の “T” をとって，これで以後の曲の演奏される速さを指定することができます．そして，前節で書いてしまったように，ディフォルトは120ですから，1分間に4分音符で120個分の速さで演奏します．そして，コンピュータですからこれは非常に正確です．次のプログラムを入力，実行して音の出ている時間を計ってみてください．

```
100 FOR I=1 TO 120
110 PLAY "C"
120 NEXT I
```



テンポを変えるとどうなるのか，実際に試しておきましょう．次の2つを比べてみてください．

```
PLAY "T60L4CDE"
PLAY "T180L4CDE"
```

どうです．同じ4分音符で演奏したわけですが，後のほうがずっと速いですね．では，もう1度，

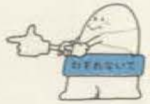
```
PLAY "CDE"
```

と打ちこんでください．後のほうと同じですね．前にやった“O”や“L”と同じく，“T”も1度セットされると，変更されるか，STOP キーやエラーでディフォルトにセットされなおすまで，それを覚えています．

さて，テンポの範囲はどこからどこまででしょうか．

```
PLAY "T0C"
```

エラーですね．この“0”を5ずつ増やしていきましょう．どうです．いくつで音が出ましたか．“35”のときですね．こんどは1ずつ減らしていくと，“31”でエラーがでました．ということは，1番遅いのは“T32”ですね．同じようにして，1番速いのをみつけてください．



正解は，“T255”です．

さて，音楽で重要なことは曲に強弱をつけるということです．きみも人と話をしているときに，だいじなことは自然に大きな声で話していますね．音楽でも同じように盛りあがりが必要なのです．

また，各パートによっても音の大きさは変化しなければなりません．音楽には，メロディーを奏でるパートとそれをひきたてるためのパートがあります．それが，逆のあるいは同じ大きさで演奏されたのでは，音楽としての全体のバランスがくずれてしまいます．

PCのPLAY命令は，同時に3つの音，すなわち3つのパートを演奏できますから，曲の流れにしたがって，しかも各パートのバランスをよく考えて演奏することが必要です．

全体の音の大きさは，PCの背中のボリュームで変えることができますが，パートごとの音の大きさは，これでは変えられません．



そのために，PLAY命令では音の大きさ（Volume）を"V"で変えることができます．やってみましょう．
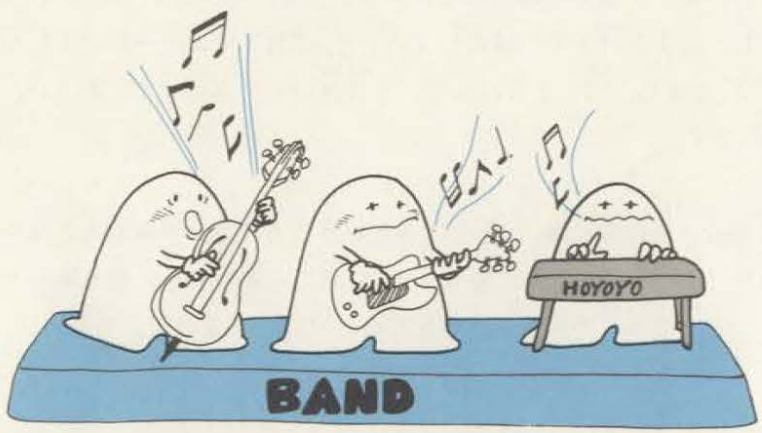
```
PLAY "V10C","V7E","V4G"
```

とキーインすると，"ド"の音が大きく，"ミ"，"ソ"の順に小さい音になっているのがわかりましたか．次は，

```
PLAY "V0C"
```

と打ちこんでみましょう．聞こえましたか．聞こえなくて当然ですよ．音は出てないのですから．この"0"を1，2，3，……の順でしだいに大きくしていってください．

どうです．だんだん大きな音になってきましたね．いくつまでいきましたか．そう，"V16"としたところで，



```
?FC Error
```

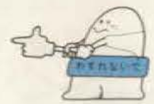


になりましたね．“V”の後ろには，0から15までの数字が付けられます．そして，ディフォルトは8です．

さて，せっかくの名演奏ができあがりになっても，PCのないところでは人に聞かせることができません．カセットテープなどに，きれいに録音できればいいですね．

PCには，このための音声出力ジャックが背中に用意されています．これに，ピンプラグのケーブルを差しこんで，もう一方をカセットやアンプにつなげばよいです．

さて，今まで出てきた“O”や“L”，“V”，そして”A”から“G”や“+”，“−”などを，PLAY命令のMusic Macro Language (MML)といいます．



MMLには，他に，“S”，“M”，“N”の3つがあります．それを説明しましょう．

まず，“S”は音の出方を変化させます．とにかく，やってみましょう．

```
PLAY "S0CCC"
```

どうです．今までとは違い，“ド”の音が3回でましたね．元に戻すには，STOPキーを押すか，

```
PLAY "V"
```

を実行します．つまり，“S”と“V”は一方の指定しか受けないのです．ためしに，次の2つを入力してみてください．

```
PLAY "V8S0CDE"
PLAY "S0V8CDE"
```



違いはわかりましたね．それでは，

```
PLAY "S0L2CDE"
```

とキーインしておいて，“0”を1，2，3，……としだいに大きくしてみてください．どうです．

音がどんどん変わっていきましたね．そして，"S16" のときにエラーが出ました．"S" には，0から15までの数字が付けられるのです．



次に "M" を説明します．この "M" は "S" とペアで使います．それでは，

```
PLAY "S10M100L2CDE"
```

と打ちこんでみてください．さきほどの音より，音の揺れ（トレモロ）が速くなりましたね．"M" はこのように揺れの速さを換える MML なのです．今までと同じようにして，"M" の後ろの数字の範囲を調べておいてくださいね．

PLAY 命令の "S" そして "M" は，"V" と同時には使えないことを，よく覚えておいてください．



さて，それでは最後の MML，"N" とまいりましょう．"N" は，Numeric Note，つまり，数字で表す音程という意味です．今まで音程は，"A" から "G" のアルファベットで表していましたが，これをすべて数字で表すためのものです．とにかく試してみましょう．

```
PLAY "N36"
```

これは，

```
PLAY "O4C"
```

と同じですね．1オクターブは，12の音からなっていましたから，36に12をたした48をやってみましょう．

```
PLAY "N48"
```

これは，さっきのちょうど1オクターブ上の "C" と同じですね．さあ，36から48の間，そしてその上や下，とにかくどんどん試してみましょう．

すると，次のような表ができますね．

O3			O4												O5					
A	A+ B−	B	C	C+ D−	D	D+ E−	E	F	F+ G−	G	G+ A−	A	A+ B−	B	C	C+ D−	D	D+ E−	E	F
33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	51	52	53

このように，"N" では1つ違うと半音違うわけです．では，1番低い音はいくつでしょう．また高い音はいくつでしょう．"N0" でも，エラーにはなりませんが，音も出ませんね．だから，"N1" が1番低い音です．高いほうは，"N96" でしたね．"N" の後ろには，1から96までの数字が付けられます．

それでは，長さはどうすれば指定できるのでしょう．そういえば，"L" という MML がありましたね．これを使ってみましょう．

```
PLAY "L16N36L4N48"
```

たしかに，長さは変りました．しかし，音程をアルファベットで表すときのように，その音だけの長さを変えることはできないでしょうか．

もちろん，できます．ただし，アルファベットのときのように続けて数字を書いたのでは，何が何だかわからなくなってしまいます．たとえば，"N364" では，4分音符なのか，64分音符なのかわかりませんね．そこで，";" を使って区別します．つまり，今のが4分音符なら，"N36；4"，64分音符なら "N3；64" とすればよいのです．やってみてください．



11−3 波の音は出ませんか

さて，PC で音を出す命令には，PLAY の他にもう1つ，SOUND というのがあります．これもながながと書くより，とにかく音を出してみましょう．ただし，こちらは PLAY 命令のように簡単にはいきませんから，覚悟はしといてくださいね．まず，




```
SOUND 7,7
```

と打ちこんでください．これだけでは，何の音も出ませんが，大切な準備をします．次に，

```
SOUND 8,8
```

とキーインすると，“ザー”という音が出ましたね．この音を止めるには，STOP キーを押します．止まりましたね．

しかし，いちいち音を出すたびに，STOP キーを押していたのではたまりません．思ったとおりに自動的に止まってくれなくては，プログラムには使えません．

それでは，もう１度音を出してください．今と同じように打ちこめばいいのです．音が出ましたね．そこで，次のように打ちこんでください．

```
SOUND 8,0
```

音が止まりましたね．

このように，SOUND 命令にはカンマで区切った，２つの数が必要です．もし１つだとどうなるでしょう．

```
SOUND 8
```

どうなりましたか？




```
?SN Error
```

になりましたね．３つだとどうなるかは試しておいてください．



SOUND 命令で必要な２つの数のうち，最初の数を“レジスタ番号”，後ろの数を“データ”と呼びます．そして，“SOUND R, D”のようにすることを，“レジスタRにDを入れる”といいます．たとえば，SOUND 8, 10は，レジスタ8に10を入れるといったぐあいです．

そして，レジスタは0から15まで，データは0から255までの数です．ですから，

```
SOUND -1,8
SOUND 7,256
```

などは，

```
?FC Error
```



となります．

さて，どのレジスタにでも，0から255までのデータを入れられるかというと，実はそうではないのです．エラーにこそなりませんが，思ったとおりにならなくなってしまいます．どのレジスタにどんなデータを入れると，どうなるかといった規則があるのです．難しいことはこれくらいにしておいて，次のプログラムを実行してみてください．




```
100 SOUND 7,7
110 FOR I=0 TO 15
120 PRINT I:SOUND 8,I
130 NEXT I
140 FOR I=15 TO 0 STEP -1
150 PRINT I:SOUND 8,I
160 NEXT I
170 END
```

どうです．波の音というには，ちょっとおかしな音ですが，そう聞こえなくもないですね．このプログラムのポイントは，レジスタ8に入れるデータを0から15，15から0と変えているところにあります．そうです．PLAY命令の“V”に似ていますね．では，レジスタ8を，レジスタ9に変えてみてください．つまり，

```
120 PRINT I:SOUND 9,I
150 PRINT I:SOUND 9,I
```

とすればいいんです．RUNすると，さっきと同じですね．では，さらに同じようにしてレジスタ9をレジスタ10に変えてください．どうですか．また同じですね．つまり，レジスタの8から10までは同じ働きをするものなのです．

PLAY命令では，3つの音が同時に出て，それぞれの音の大きさが別々に変えられたのを思い出

してください．実は，このレジスタ8，9，10がこのそれぞれのパートの音の大きさをコントロールしていたのです．そして，エラーを起こしたり，STOPキーを押したりすると，PCは自動的にこれらを全部0にセットするわけです．



さて，話は戻りますが，ここで出てきた“ザー”という音をノイズといいます．ノイズは，ゲームなどには非常に便利なもので，たとえば，波の音やミサイルの爆発音，ジェット機やレーシングカーの音などに使えます．それでは，次のプログラムを実行してみてください．

```
100 SOUND 7,7:SOUND 8,8
110 FOR I=0 TO 31
120 PRINT I:SOUND 6,I
130 FOR J=1 TO 50:NEXT J
140 NEXT I
150 END
```

新しく，レジスタ6が仲間に加わりました．そして，ノイズの感じが変わっていったのがわかるでしょう．音の大きさも変わっているように聞こえますが，レジスタ8は変えていません．

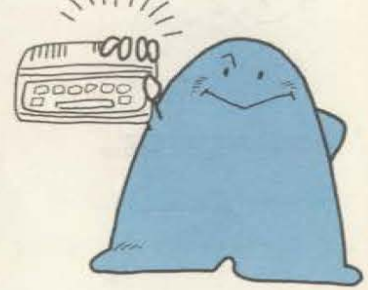
レジスタ6は，このようにノイズの感じを変えるもので，0から31まで変えることができます．では，それ以上はどうなるでしょう．次のようにプログラムを変更して，実行してみてください．

```
110 FOR I=0 TO 100
```

どうでしたか．32以上は，0から31までの繰り返しでしたね．ですから，レジスタ6は0から31まで変えれば，それで充分なわけです．

では，次へまいりましょう．レジスタの8，9，10は音の大きさを変えるものでしたが，さきほどは0から15までしか変えませんでしたね．今度は16を入れてみます．次のプログラムを実行してみてください．

```
100 SOUND 7,7:SOUND 8,16
110 FOR I=0 TO 31
120 PRINT I:SOUND 13,I
130 FOR J=1 TO 300:NEXT J
140 NEXT I
150 SOUND 8,0
160 END
```



今度は，レジスタ13に順にデータを入れています．おもしろい音になるのは，8から15の間ですね．24から31まではこの繰り返しだったのがわかりましたか．どんどん続けましょう．次はこのプログラムです．

```
100 SOUND 7,7:SOUND 8,16:SOUND 1
3,8
110 FOR I=0 TO 31
120 PRINT I:SOUND 12,I
130 FOR J=1 TO 500:NEXT J
140 NEXT I
150 SOUND 8,0
160 END
```

最初は機関銃が，最後は蒸気機関車のような音になりましたね．レジスタ13のデータを他の8から15の数に変えてみてください．ヘリコプターのような音もありますよ．

また，少し変えます．次のプログラムです．

```
100 SOUND 7,7:SOUND 8,16:SOUND 1
3,8:SOUND 12,1
110 FOR I=0 TO 255 STEP 8
120 PRINT I:SOUND 11,I
130 FOR J=1 TO 500:NEXT J
140 NEXT I
150 SOUND 8,0
160 END
```

どうです．さっきより少しずつ変わっていきましたね．

それでは，まとめてみましょう．まず，レジスタ8を16にすると音量を変えるのではなく，音が揺れるようになります．これは，レジスタ9でも10でも同じです．この音の揺れ方はレジスタ13で変えることができます．そして，この揺れの速さをレジスタ12で大きく，レジスタ11で細かく変えることができます．



11－4　ピアノの音？オルガンの音！

SOUND命令はなにもノイズを出すためだけにあるものではありません．きれいな音も自由自在に操ることができますし，曲を演奏することももちろんできます．PLAY命令よりも自然な音，つまり楽器に近い音を出すことができますが，その分難しいのでがんばって理解してください．

では，次のプログラムを実行してみてください．

```
100 SOUND 7,56
110 FOR I=0 TO 15
120 FOR J=0 TO 255
130 PRINT I,J
140 SOUND 1,I:SOUND 0,J
150 SOUND 8,8
160 NEXT J,I
170 END
```

PLAY命令よりも音の高さ，つまり周波数が細かく変わっていきましたね．リストからもわかるように，レジスタ0に0から255，レジスタ1に0から15を入れていきます．そして，周波数はレジスタ0を変えると少し，レジスタ1を変えると大きく変化していますよね．わからなかった人は，もう1度画面をよく見ながら，しっかり聞いてください．

このように，レジスタ0と1は2つ1組で音の高さを決める働きがあります．ところで，PLAY命令では3つの音を同時に出すことができました．その1つが，レジスタ0と1だったとしたら，あと2組同じようなレジスタがあるはずです．

では，さっきのプログラムのレジスタ0を2に，レジスタ1を3に変えてみましょう．次のように打

ちこんでから，実行してください．

```
140 SOUND 3,I:SOUND 2,J
```

おや，周波数は変わりませんね．この音はさっきの1番最後に出ていた音ではありませんか．これはどうしてなのでしょう．もう1度，プログラムをよく見てみましょう．

わかりましたか．そう，レジスタ8をレジスタ9に変えなければならないのですね．では，やってみましょう．

```
150 SOUND 9,8
```

と打ちこんでから，実行してください．今度はさきほどと同じようになりましたね．同じようにして，レジスタ2を4に，レジスタ3を5に，そして忘れずにレジスタ9を10に変えてやってみてください．これも同じはずです．

では，レジスタ4を6に，レジスタ5を7に，レジスタ10を11に変えられるでしょうか．わからない人は，前節をもう1度よく読んでおいてください．

それでは，まとめてみましょう．レジスタは，0と1，2と3，4と5で2つ1組になって音の高さを決めています．そして，それぞれ，8，9，10でその大きさを変えられるのです．もちろん，同時に別々の高さ，大きさで音を出すことができますよ．次のプログラムで確かめておきましょう．

```
100 SOUND 7,56
110 SOUND 0,0:SOUND 1,1
120 SOUND 8,7
130 FOR I=1 TO 100:NEXT I
140 SOUND 2,128:SOUND 3,1
150 SOUND 9,9
160 FOR I=1 TO 100:NEXT I
170 SOUND 4,128:SOUND 5,0
180 SOUND 10,12
190 FOR I=1 TO 500:NEXT I
200 SOUND 8,0:SOUND 9,0
210 SOUND 10,0:END
```

きれいな音がでることはわかりましたが，音階とレジスタに入れるデータは，どのような関係になっているのでしょうか．これがわかれば，曲を演奏することもできます．そこで，次のプログラムでPLAY命令の“O4C”に当たるデータを探しましょう．レジスタ4に0〜255の数，レジスタ5に0〜15の数を入れています．

```
100 SOUND 7,56
110 PLAY "O4C1"
120 INPUT "Sound 4-";S4
130 INPUT "Sound 5-";S5
140 SOUND 4,S4:SOUND 5,S5
150 SOUND 10,8:GOTO 110
```



どうなりましたか．コツは，始めはレジスタ4は0にしておいて，レジスタ5を変えてだいたい合わせてから，レジスタ4を少しずつ変えていくことです．

それでは，正解を発表しましょう．レジスタ4は221で，レジスタ5は1です．わからなかった人は試してみてください．

次は，オクターブ上の“O5C”についてやってみましょう．次の部分を変更して試してください．

```
110 PLAY "O5C1"
```

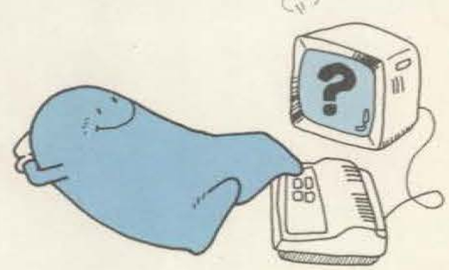
どうですか．今度は，レジスタ4は239，レジスタ5は0ですね．

さて，1オクターブ違う音というのは，周波数ではちょうど2倍，または2分の1の関係になっています．そして，オクターブ内の音はたがいに2の12分の1乗の比になっています．ですから，さきほど求めたデータからすべての音程のデータを求めることができるのです．

そこで，さきほどのデータをよく見てください．一方は1と221，もう一方は0と239です．レジスタ4は0から255で，周波数はレジスタ5を変えたときより細かく変わるのでしたね．レジスタ4に256を入れたいときに，かわりにレジスタ5に1を入れると考えると，このデータは，一方が256＋221で477になって，もう一方の約2倍になっています．これを応用すれば，他の音程のデータも実際に耳で1つ1つ確かめなくても，計算で得られるのではないでしょうか．

では，次のプログラムを実行してみてください．

```
100 C=477:SOUND 7,59:SOUND 10,0
110 FOR I=12 TO 0 STEP -1
120 D=INT(C*2^(I/12)+.5)
130 PRINT D
140 H=INT(D/256):SOUND 5,H
150 L=D-H*256:SOUND 4,L
160 SOUND 10,8
170 FOR J=1 TO 100:NEXT J
180 SOUND 10,0:NEXT I:END
```



どうやら，うまく音階になったようですね．
それでは，次の2つのプログラムを実行して聞き比べてみてください．



```
100 SOUND 7,59:SOUND 10,0
110 FOR N=1 TO 8
120 READ H,L
130 SOUND 4,L:SOUND 5,H
140 FOR I=0 TO 15 STEP 5
150 SOUND 10,I
170 NEXT I
180 FOR J=1 TO 5:NEXT J
190 FOR I=15 TO 0 STEP -1
200 SOUND 10,I
220 NEXT I,N
230 END
240 DATA 3,186,3,82,2,245,2,203
250 DATA 2,125,2,55,1,249,1,221
```




```
100 SOUND 7,59:SOUND 10,0
110 FOR N=1 TO 8
120 READ H,L
130 SOUND 4,L:SOUND 5,H
140 FOR I=0 TO 15 STEP 5
150 SOUND 10,I
170 NEXT I
180 FOR J=1 TO 100:NEXT J
190 FOR I=15 TO 0 STEP -5
200 SOUND 10,I
210 FOR J=1 TO 5:NEXT J
220 NEXT I,N
230 END
240 DATA 3,186,3,82,2,245,2,203
250 DATA 2,125,2,55,1,249,1,221
```

どうです．片方はピアノのような感じに，もう片方はオルガンのような感じに聞こえませんでしたか．詳しくは説明しませんが，違いはFOR文の部分だけです．きみもいろいろと試してみてください．

11－5　まさかドラムの音は？

これで，SOUND命令のうち残レジスタは7，14，そして15の3つとなりましたが，このうちレジスタ14と15は，音とは無関係ですから後はレジスタ7だけです．

レジスタ7は，今までにもおまじないのように使ってきましたが，それだけ重要な役割をするということなのです．

さて，レジスタ8，9，10は音の大きさをコントロールするレジスタでしたね．そして，今までいろいろと試してきたように，あるときはレジスタ0・1，2・3，4・5で決められた高さの音の大きさを，またあるときはノイズの大きさを変えてきました．

つまり，同じレジスタなのにきれいな音とノイズの両方の大きさを変えることができるわけです．そしてそのどちらを使うかを決めるのが，このレジスタ7なのです．

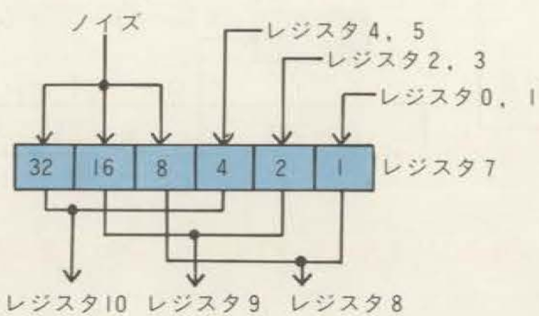
次のプログラムを実行してください．




```
100 SOUND 0,0:SOUND 1,1
110 SOUND 2,128:SOUND 3,1
120 SOUND 4,128:SOUND 5,0
130 SOUND 8,6:SOUND 9,9
140 SOUND 10,12:SOUND 7,63
150 INPUT "ﾚｼﾞｽﾀ 7 ﾊ ";R
160 SOUND 7,R
170 GOTO 150
```

PC が，“レジスタ7は”と聞いてきましたね．まず，“56”と答えてください．きれいな音ですが，少しずつ周波数を変えてありますからうなりが聞こえますね．しかし，ノイズは出ていません．次は“7”を入力します．今度はノイズだけになりましたね．そして，“0”です．今度はノイズときれいな音がまざっていますね．最後は，“63”です．音は消えました．他の数もいろいろと試しておいてください．

では下の図を見てください．



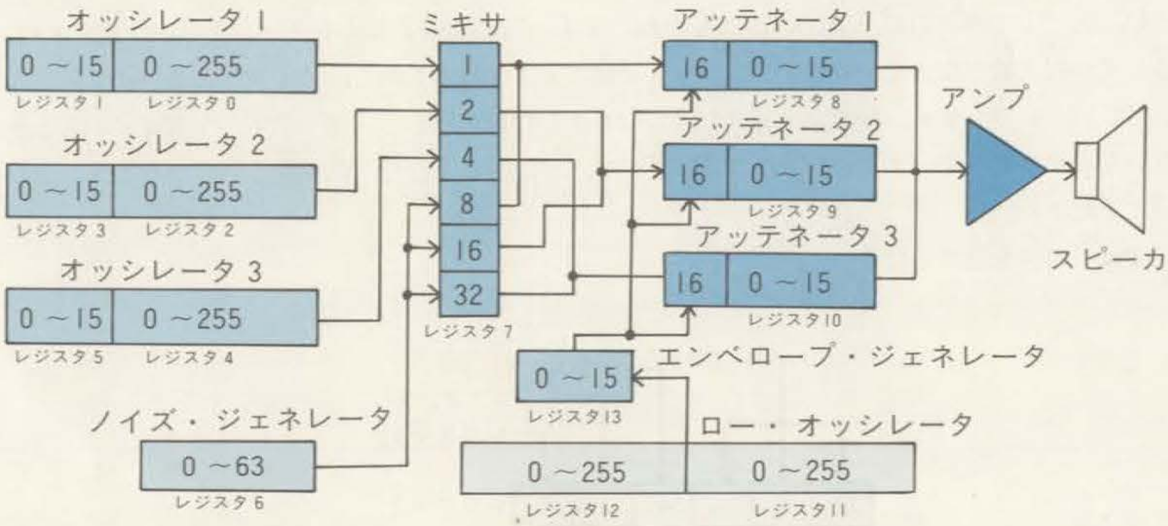
これは，レジスタ7に入れるデータと他のレジスタの関係を示しています．そして，レジスタ7の中の数がどの音を出さないかを決めるものです．

ちょっと難しそうに見えますが，たとえば，さきほどのプログラムで最初に56を入れましたが，これは“8+16+32”，つまりノイズを出さないデータなのです．そして次に7を入れましたが，これは“1+2+4”で，きれいな音を出さないデータということです．もちろん，0のときは全部出す，

63のときは“1+2+4+8+16+32”で全部出さない，となるわけです．

どうです．レジスタ7はわかりましたか．さっきのプログラムをもう1度実行して，今度はこの図を見ながらいろいろなデータを入れてみてください．思いどおりになるはずです．

これで，SOUND命令は全部終わりました．次の図は，レジスタをまとめたものです．シンセサイザを操ったことのある人はすぐわかるでしょうが，これはまさにシンセサイザなのです．



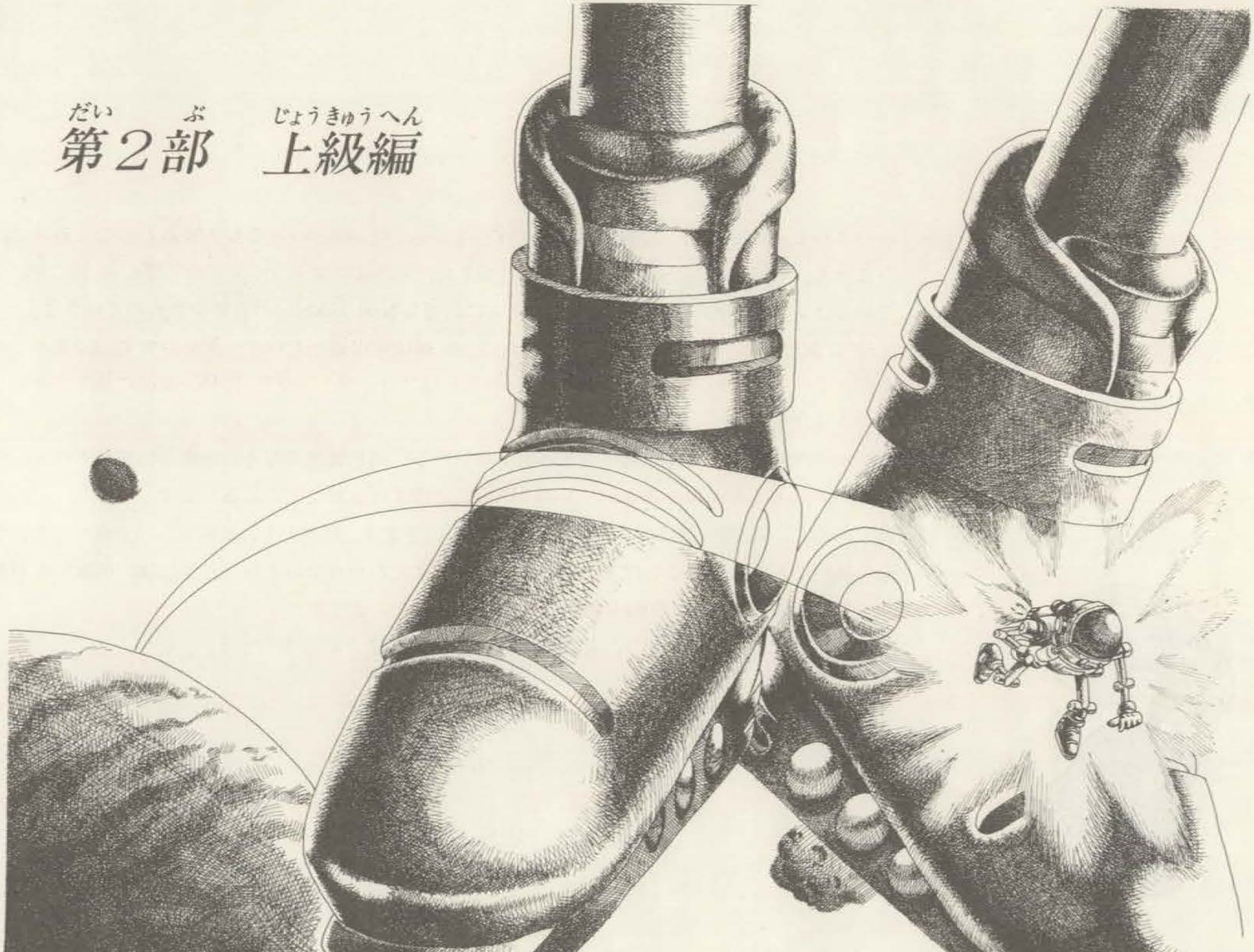


では，最後にドラムのような音（？）を作ってみましょう．プログラムを見る前に，きみも工夫してやってみてください．



```
100 SOUND 4,0:SOUND 5,11
110 SOUND 6,0:SOUND 7,27
120 SOUND 10,8:SOUND 10,15
130 FOR I=15 TO 0 STEP -3
140 SOUND 10,I
150 IF I<14 THEN SOUND 7,59
160 NEXT I:SOUND 10,0
170 END
```

第2部　上級編



いよいよ，第2部は上級編の始まりです．しかし，上級といっても突然難しくなるわけではありません．第1部の初級編をしっかりやってきた，きみはプログラミングの基礎が充分に身についているはずです．その基礎を忘れずに，さらに高等なN60-BASICの世界をきわめていきましょう．

初級編がいわゆる基礎編というならば，上級編は実用編ともいうべきものです．第2部のこの扉を開けようとしているきみなら，たいていのことはプログラミングして，PC-6001に処理させることができるはずです．

しかし，もっと高度なことや美しい表示をしたり，同じ処理でももっと簡単にプログラミングするための，とても便利な命令が，N60-BASICにはまだまだたくさんあります．

もちろん，第2部の話は少しずつ難しくなっていきます．けれども，基礎をしっかりやった，きみなら絶対にわからないことはありません．どうもよくわからないというときには，何度でも自分でやってみることが，PC-6001を使いこなすには大事なことです．

その努力を忘れずに！では，第2部上級編を始めることにしましょう．



第1章　スクリーン・コントロール



第2部の先頭をきって現れるのが，このスクリーン・コントロールの命令群です．まず実用になるプログラムを作るためには，何はさておき，画面に美しく表示することが大切です．

画面の表示を思いどおりにする．それが，スクリーン・コントロールなのです．N60-BASICには，このスクリーン・コントロールのためのとても便利な命令がいくつかあります．

それほど数は多くありませんから，1つ1つを完全にマスターしてくださいね．スクリーン・コントロールの命令は，使いこむほど奥の深いものばかりですから，命令の1つ1つをよく知ることが大切です．

1－1　じゃまものは消せ！

画面最下行のファンクション・キー表示と，キーを押したときの，"ポッ！ポッ！"というクリック音，どう思いますか．ゲームの際などには，じゃまになりそうですね．なんとかできないものでしょうか．

じゃまものには消えてもらいましょう．そのための命令が，"CONSOLE"です．

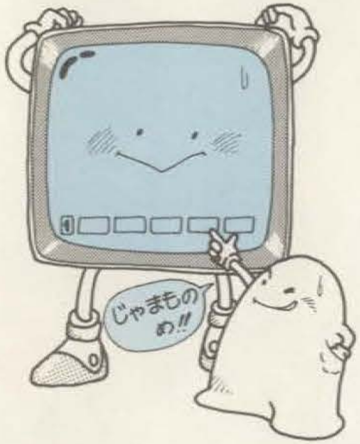


では，まず次のように打ちこんでみてください．

```
CONSOLE ,,0
```

ほら，今まで出ていたファンクション・キー表示が消えましたね．では，今度は，

```
CONSOLE ,,1
```



とキーインしてみてください．ふたたび，ファンクション・キー表示が現れました．

このように，CONSOLE命令の3番目のパラメータ(1番目，2番目はカンマの前にあるはずですが，省略してこのように打ちこめます)を，1または0にすることにより，ファンクション・キー表示をきみの思いどおりにすることができます．

それでは，0や1以外ではどうなるのでしょうか．やってみましょう．たとえば，

```
CONSOLE ,,2
```

と打ちこんでみましょう．

```
?FC Error
```

となってしまいましたね．他の数でも同様にエラーになります．

つまり，CONSOLE命令の第3のパラメータは，0と1しか受けつけないということです．

ということは，どうやら，クリック音の操作は第4のパラメータらしいですね．では，次のように打ちこんでみましょう．

```
CONSOLE ,,,0
```

そして，何かキーボードをたたいてみましょう．どうですか．さっきまでのキーを押したときのクリック音はなくなりました．第4のパラメータも第3のパラメータと使い方が似ているようです．では，今度は，

```
CONSOLE ,,,1
```

と打ちこんでみることにしましょう．ふたたび，何かキーボードをたたいてみましょう．キーを押したときのクリック音が，また出るようになりました．



このように，CONSOLE命令の4番目のパラメータを，0または1にすることにより，キーを押したときのクリック音を思いどおりにすることができます．

それでは，0や1以外ではどうなるのでしょうか．たとえば，

```
CONSOLE ,,,-3
```

と打ちこんでみましょう．やっぱり，

```
?FC Error
```

となってしまいました．このパラメータの場合も，他の数では同様のエラーになります．

つまり，CONSOLE命令の第4のパラメータも，0と1しか受けつけないということです．

1－2　スクロール・ウィンドウ

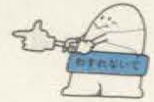
実は，このCONSOLEという命令は前節のような，ファンクション・キーの表示やキーのクリック音を出したり，止めたりするためのものではありません．

前節では，省略していた第1パラメータおよび第2パラメータを使って初めて，CONSOLEの機能が生きてくるのです．では，それらを説明するまえに，実際にやってみましょう．まず，画面をクリアして，次のように打ちこんでください．なお，第3・第4パラメータは特に指定しませんから，きみの好きなように設定しておいてください．



```
CONSOLE 5,5
```

これだけでは何も起こりませんが，カーソルを下におろしてみてください．画面の上から3分の2ほどおりたところで止まってしまいました．では，上にあげてみて下さい．上から3分の1ほどの所で止まってしまいましたね．

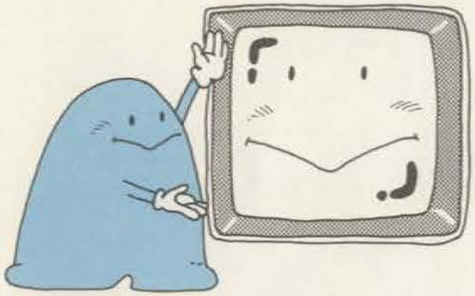


実は，このCONSOLEという命令はこのように，画面の好きな行から好きな行数の部分にだけ表示を行うためのものなのです．この，好きな部分というのを“スクロール・ウィンドウ”といいます．

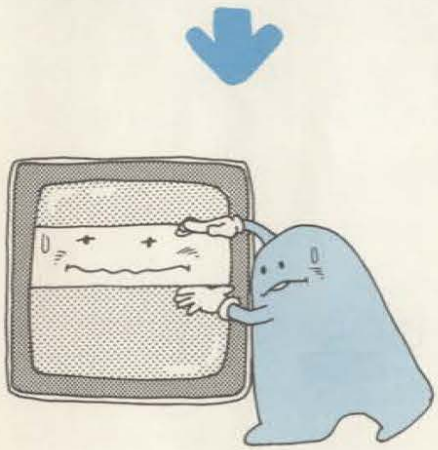
このスクロール・ウィンドウを設定すると，PRINT文などにより表示され，自動復帰改行によりスクロールされる部分は，そこだけに限られてしまいます．つまり，CLS命令やクリアキーはこの部分だけを消去し，他の部分は変化しません．

また，設定されたスクロール・ウィンドウの左上が新たなホーム・ポジションとなります．そして，1度このスクロール・ウィンドウの中へ入ってしまったカーソルは，カーソル移動キーでは外へ出すことができません．

では，次のサンプル・プログラムを入力して，実行させてみてください．画面から目をはなさないように！



```
100 FOR I=0 TO 15
110 FOR J=1 TO 30
120 CONSOLE I,16-I
130 PRINT J
140 NEXT J,I
150 END
```



どうです．上から順に表示されている所が下がってきたでしょう．しかも今は，画面の1番下の1行にスクロール・ウィンドウが設定されています．試しに，リストを取ってみてください．見えましたか？

このように，CONSOLE命令では，第1のパラメータでスクロール・ウィンドウの開始行を，第2のパラメータでスクロール・ウィンドウの行数を指定できます．

このままでは不便ですから画面を元の状態に戻しましょう．そのためには，開始行は1番上，行数は16行ですから，

```
CONSOLE 0,16
```

と入力してください．

第1のパラメータは，1番上が0，1番下が15です．それ以外の場合，どうなるかは今までと同じように，ぜひ試しておいてください．

次の第2のパラメータは，第1のパラメータで示された行からの行数ですから，1から16までですね．

さて，このCONSOLE命令のうまい使い方は，消したくない部分を避けてスクロール・ウィンドウを設定するようにする方法です．特にゲームなどでは，遊ぶ人は必ずしもコンピュータの使い方

を良く知っているとは限らないのですから，入力ミスなども多いものです．

入力ミスで，せっかくの画面がスクロールしてしまって見えなくなってしまっては，ゲームの興味がうすれてしまいますね．

そこで，次のゲーム盤のように盤を作ってしまってから，この命令で盤自体はスクロールしないようにしてしまえばいいのです．

```
100 CONSOLE 0,16:CLS
110 A$="   1 2 3 4 5 6 7 8"
120 B$=" + + + + + + + +"
130 PRINT A$
140 FOR I=1 TO 8
150 PRINT I;B$;I
160 NEXT I
170 PRINT A$
180 CONSOLE 11,5:CLS
190 END
```

最後に，CONSOLE 命令のパラメータは，第1と第2はペアで，第3，第4はそれぞれ独立しています．これらのパラメータを，たとえば次のように省略した場合には，基本的にはそれ以前に設定されたものになります．

```
CONSOLE 5,8
CONSOLE ,,1,0
CONSOLE ,,,1
```

などいろいろ試してみてくださいね．ただし，第1のパラメータと第2のパラメータはそれぞれ関わりあっているわけですから，一方だけを省略することは避けるべきです．特に第2パラメータの省略は，エラーになったり，思わぬところにスクロール・ウィンドウが設定されてしまったりしますから，なるべくなら，省略しない方が良いでしょう．

たとえば，次の例をやってみましょう．

```
CONSOLE 0,3
```

としてから，カーソルを上下に動かしてみてください．そして次に，

```
CONSOLE 5
```

を実行すると………．ほら，



```
?FC Error
```

が出ましたね．

ところが，第2パラメータを省略すると，常にエラーになるというわけではありません．では，まず，

```
CONSOLE 5,3
```

として，カーソルを上下に動かして，さっきのように，どこがスクロール・ウィンドウになっているのか確かめておいてください．そして次に，

```
CONSOLE 0
```

を実行すると………．今度は，エラーは起こりませんでしたね．しかし，カーソルを動かしてみると，3行だけのはずのスクロール・ウィンドウですが，違っています．

このように，エラーを起こすときと，そうでないときがあるために，第2パラメータだけは省略しない方がよいのです．

1－3　好きなところに表示するには

動きのあるゲームや，きれいな表を作ろうと思ったとしましょう．そのとき，必要となるのが，画面の好きな場所に文字や数字を表示させるテクニックです．

ダイレクト・モードでは，カーソル移動キーがありますから，画面の好きな位置にカーソルを移動させて，そこに文字や数字を書くことができます．けれども，プログラム実行中に，カーソル移動キーを用いることはできません．



そこで，N60-BASICには，そのための命令もちゃんと用意されています．それが，LOCATE命令なのです．これを使えば，画面上の好きな位置にカーソルを移動させることができます．ただし，カーソルは表示されません．

まずは，とにかく使ってみましょう．画面をクリアしてから，次のように打ちこんでください．

```
LOCATE 9,3
```

すると，上から6行目にカーソルが表示されました．これでLOCATE命令により，縦方向にカーソルが移動したことはわかりましたが，横方向はどうなっているのでしょう．

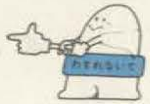
そうです．カーソルは次の文字が表示されるべき場所を示すものであったことを思い出してください．そこで，次のように打ちこんでみましょう．もちろん，画面をクリアしてから行ってください．

```
LOCATE 9,3:? "A"
```

するとどうでしょう！4行目の左から10文字目の所にAの文字が出ました．このように，LOCATE命令は，PRINTとペアで使うものなのです．

ところで，もう1度，入力したものと出てきた位置を見比べてみてください．9，3と指定したのに，4行目の10文字目に出ていますね．どうやら，最初のパラメータが横方向の位置を，そして次のパラメータが縦方向の位置を，それぞれ指定しているようです．

しかも，指定した値よりもそれぞれ1つずつ多くなっているということは，最上行と左端が0であるということです．そうすると，9，3と指定したのに10，4に表示されたわけがわかります．



つまり，画面の左端が0ならば右端は31になり，画面の最上行が0ならば最下行は15であるということです．ただし，特にSCREEN3のロー・レゾリューション・モードの場合，画面の横サイズがその他のモードの半分ですから，横が15までになります．

いずれの場合にしても，とにかく，9や3だけではなく，いろいろと数値を変えて試してみてください．

それでは，この命令を用いて何かプログラムをつくってみましょう．とても応用範囲が広いのですが，ここでは画面上の“＊”をコントロールするプログラムを作ってみようと思います．

入力にはINKEY$を使って，操作キーは，Wで上，Zで下，Aで左，Sで右，の4つを使いましょう．

そこで，プログラムは次のようになります．




```
100 CONSOLE 0,16,0:CLS
110 H=0:V=0
120 LOCATE H,V:PRINT "*"
130 A$=INKEY$:IF A$="" THEN 130
140 LOCATE H,V:PRINT " "
150 IF A$="w" THEN V=V-1
160 IF A$="z" THEN V=V+1
170 IF A$="a" THEN H=H-1
180 IF A$="s" THEN H=H+1
190 IF V<0 THEN V=0
200 IF H<0 THEN H=0
210 GOTO 120
```

RUNさせると，画面の左上に＊が現れました．では右にずっと動かしてみましょう．

あっ！右端に来た所で，

```
?FC Error in 120
```

で止まってしまいました．なぜでしょうか．ダイレクト・モードでLOCATEのパラメータの内容を調べてみましょう．

```
PRINT H,V
```

すると，Hは32，Vは0であることがわかりました．先ほどすでに説明しましたように，画面の右端は31です．つまり，そのサイズよりもパラメータが大きくなるとエラーになるということですね．



それでは，Hが31より大きくならないようにするためのプログラムを付け加えることにしましょう．また，同じように，縦は15までですからVを15以下にしておかなくてはなりませんから，これ

もいっしょに付け加えることにしましょう.

そうすると, 次のようにすればよいでしょう.

```
205 if v>15 then v=15
206 if h>31 then h=31
```

再び, 実行してみましょう. 今度は右端に来ても大丈夫ですね. それでは下に動かしてみましょう.

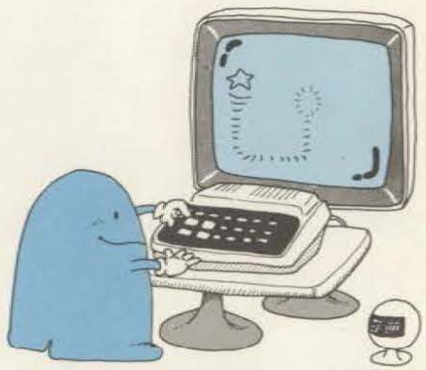
おやっ! 1番下に来たところで, エラーにはならないのに, 表示がおかしくなってしまいました. どんどん下へ行こうとすると, *が1つおきに上の方へ流れていきます. しかも, 1番下の*はチカチカとまたたいているようです. どうしてかわかりますか. 問題は "*" の表示の所ですから, 原因はPRINTにありそうですね.

そうです. PRINTの最後がカンマ (,) やセミコロン (;) でない場合は改行されてしまうのでした. プログラム中の2カ所のPRINTの最後にセミコロンを付けてください.



```
120 LOCATE H,V:PRINT "*";
140 LOCATE H,V:PRINT " ";
```

今度は完璧のようです. このように, LOCATE命令の次のPRINTの最後はセミコロンにしておくのがよいのです.



実は, このプログラムではまだ1つだけ, どうしようもないことがあるのです. すでに気づいている人もいるかもしれませんが, *が画面の右下に来た時は, どうしてもスクロールしてしまうため, 表示がおかしくなってしまいます.

これを避ける方法は, ここに*を来させないようにしてしまうことです.

```
205 IF V>14 THEN V=14
```

さて, 前節でスクロール・ウィンドウを設定して, ゲーム盤などのせっかく作った表示を壊さない方法を学びましたね. しかし, これとてずっとこのままでいいわけではありません. 例えば, ゲー

ム盤などは，次にこまを置く位置をきみが入力したら，そこにこまを表示しなければならないのです．

そこで，活躍するのが，このLOCATE命令です．この命令でカーソルを動かすと，スクロール・ウィンドウがどこであろうと，絶対そこに移動します．

実際にやってみましょう．まず，画面をクリアしてから，

```
CONSOLE 5,5
```

とキーインします．もちろん，スクロール・ウィンドウは，上から6行目から11行目までに設定されますね．カーソルを上下に動かして確かめておいてください．1度，スクロール・ウィンドウに入ったカーソルは，外へは出られません．

さて，この状態で1番上の行の6文字目に"A"を表示するには，どうしたらいいでしょう．カーソルが，そこに行けないのですから困りました．

そこで，次のように打ちこんでみてください．

```
LOCATE 6,0:? "A";
```

どうです．カーソルはみごとスクロール・ウィンドウから脱出して，思いどおりのところに"A"を表示することができました．

これを使って前節のゲーム盤に，こまを置く部分を加えましょう．

```
100 CONSOLE 0,16:CLS
110 A$="   1 2 3 4 5 6 7 8"
120 B$=" + + + + + + + +"
130 PRINT A$
140 FOR I=1 TO 8
150 PRINT I;B$;I
160 NEXT I
170 PRINT A$
180 CONSOLE 11,5:CLS
190 INPUT "どこに うちますか。(よこ)";X
200 INPUT "どこに うちますか。(たて)";Y
210 LOCATE X*2+1,Y:PRINT "o";
220 END
```

1－4　カーソルは今どこに

無事，スクロール・ウィンドウから飛び出して，壊したくない部分にも表示できるようになりました．しかし，そのままにしておいては，再び壊したくない部分に次々と表示されてしまいます．

これでは，せっかくのスクロール・ウィンドウが無駄になってしまいますね．そのためには，スクロール・ウィンドウの中から壊したくない部分にカーソルを移動する前に，その時点でのカーソルの位置を覚えておけばいいのです．



N60-BASICには，そのためにPOSとCSRLINという2つの命令が用意されています．たとえば，これらを使って，前節の“＊”を自由に動かすプログラムに，“＊”の位置を知るプログラムを付け加えてみましょう．そこで次のようなプログラムを考えてみました．

```
100 CLS:CONSOLE 15,1:CLS
110 INPUT "よこ は";H:INPUT "たて は";
V
120 LOCATE H,V:PRINT "*";
125 P=POS(0):L=CSRLIN
128 LOCATE 0,15:PRINT "よこ";P,"たて
";L;" ";
130 A$=INKEY$:IF A$="" THEN 130
140 LOCATE H,V:PRINT " ";
150 IF A$="w" THEN V=V-1
160 IF A$="z" THEN V=V+1
170 IF A$="a" THEN H=H-1
180 IF A$="s" THEN H=H+1
190 IF V<0 THEN V=0
200 IF H<0 THEN H=0
205 IF V>14 THEN V=14
206 IF H>31 THEN H=31
210 GOTO 120
```

実行させると，“よこ　は？”，“たて　は？”と，“＊”の最初の位置を聞いてきますので，適当な数字を入力してください．画面に“＊”が現れ，下にその位置が表示されています．

ところが，ちょっと待ってください．入力した位置とは違っているようです．少しあちこち動か

してみてください．

というのも，ここに示されているのはカーソルの位置であって，“＊”の位置ではないのです．つまり文字が表示されると，この場合，“＊”の表示の右側にカーソルが移動するのです．そのため，“＊”が右端に来ると，横が0になって，縦が1つ増えているのです．

POSや今までのように，ダイレクト・モードで試してみましょう．まず，引数を0にして，




```
? " ";pos(0)
```

とすると，1が表示されましたね．それでは引数を1にしてやってみてください．同じですね．実は，この関数の引数はダミーなのです．

次のCSRLINですが，これには引数はなく，一見変数のようです．ずいぶん長い変数ですね．もし変数ならば，値の代入ができるはずです．ダイレクト・モードで何か代入してみましょう．

```
csrlin=10
```

どうなりましたか．




```
?SN Error
```

となりましたね．つまりこの変数は，代入ができないのです．このような変数のことを予約変数とか特殊変数などと呼んでいます．

それでは，前節の最後で作ったゲーム盤をちゃんと動くようにしましょう．プログラムは次のようになります．

```
100 CONSOLE 0,16:CLS
110 A$="   1 2 3 4 5 6 7 8"
120 B$=" + + + + + + + +"
130 PRINT A$
140 FOR I=1 TO 8
150 PRINT I;B$;I
160 NEXT I
170 PRINT A$
```


プログラム続く



```
180 CONSOLE 11,5:CLS
190 INPUT "どこに うちますか。(よこ)";X
200 INPUT "どこに うちますか。(たて)";Y
205 H=POS(0):V=CSRLIN
210 LOCATE X*2+1,Y:PRINT "○";
220 LOCATE H,V:GOTO 190
```

このゲーム盤を使って，何をやるかはきみの腕しだいです．おもしろいゲームを作ってください．



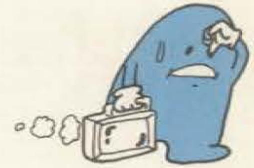
第2章　文字だって思いのまま



画面を思いどおりに扱えるようになったら，次はそこに表示するものが問題となります．数だってカッコよく表示したいし，文字を思いどおりに表示したいですね．

さて，この章では文字列を自由に切ったり，組み合わせたりする方法をお教えします．さらに，文字を数として扱う，あるいは逆に数を文字として扱うなどの方法も説明しましょう．

文字に細工をほどこして，立派な表示を作って，画面を操れるようになれば，ちょっとしたデザイナーになれますよ．



2－1　文字列を自由に操る

文字列，つまり文章をつなぐ方法は初級編の第4章でやりましたね．覚えていますか．忘れてしまっていたら，もう1度そこを読み返して，しっかり身につけておいてください．

ここでは，文字列をわけたり，一部分を抜き出したりする方法について説明しましょう．その方法というのは，特別に用意されている文字列用の関数を使う方法ですが，面倒なことではありません．逆に，それらを使い慣れると，数値を操るのよりも便利なこともあります．

さて，そのような文字列関数を使って，文字列をわけたり，抜き出したりするためには，その長さを指定してやらなくてはなりません．そこで，それらの文字列関数を使う前に，文字の長さを知る方法について説明しなくてはなりませんね．



N60-BASICには，この方法として，LEN関数が用意されています．では，これを使って，名まえの長さを知るプログラムを作ってみましょう．とても簡単なプログラムですから，きみにも作れると思います．それは次のようになります．

```
100 INPUT "あなたの おなまえは";N$
200 PRINT "'";N$;"' は";LEN(N$);"
 もし゛ て゛す。"
300 END
```

RUN させてみましょう．もし，きみの名前が，山田太郎君なら，かなモードにして"やまた゛たろう"と入力することになります．すると，LEN 関数によって，きみの名まえが7文字であることがわかりました．ふつう，濁点は1文字と数えませんが，キーボードでは濁点は1つの文字として独立していますので，1文字と数えます．

では，もう1度 RUN させて，今度は，名字と名前の間に空白をいれてみましょう．どうなるでしょうか．

さっきより，1文字多くなりましたね．つまり，空白も1つの文字とみなされているわけです．このように，LEN 関数は，空白や，文字でない文字も数に含んでいます．

では，もし引数がヌルストリングだったらどうなるでしょう．それは，




```
? LEN("")
```

で確かめておきましょう．どうなりましたか．そうです．0ですね．ヌルストリングは長さゼロの文字列ですから，当然といえば当然ですね．

LEN 関数は文字列の長さを知るものですから，ここに普通の変数や数などを入れるとエラーになりますよ．やっておきましょう．

```
? LEN(a)
```

とすれば，




```
?TM Error
```

ほらね！

文字列の長さを知る方法についてわかったところで，文字列をわける方法について話を進めま

しょう．

まず，さっきのプログラムを使って，あなたの名前の頭文字を抜き出す方法から始めましょう．

では，200行を次のように変えてみてください．

```
200 PRINT "かしらもじ は '";LEFT$(N$,
1);"' です。"
```



また，違う関数が出てきました．実行させてみるとわかるとおり，この関数は文字列の先頭から（左から），任意の長さの文字列を取り出す関数なのです．この場合は，第2引数が1なので，1文字だけを抜き出しているわけです．これを変えることによって，好きなだけの長さの文字を抜き出せます．

では，第2引数を0にしてみたらどうなるでしょうか．試しにやってみましょう．

```
? LEFT$("YMO",0)
```

第2引数は，さっきいったとおり，抜き出す文字列の長さを指定するものなので，当然ヌルストリングとなり，何も表示されないのです．

では，第1引数の文字列の長さよりも長い値にした場合にはどうなるでしょう．これもやってみましょう．

```
? LEFT$("YMO",4)
```

エラーにならず，第1引数の文字列全てを表示しましたね．

さて，左からとる関数があるなら，右からとる関数もあるだろうと気づいた人は，なかなか頭がいいですね．その通りあるのです．それはRIGHT$関数というものですが，LEFT$の文法と同じですから，今度は自分でやっておいてくださいね．

LEFT$関数は左から，RIGHT$関数は右から，それぞれ文字列を抜き出しますが，では，文字列のまん中を抜き出すにはどうしたらいいでしょう．たとえば，




```
"わたしは PC-6001 です"
```

という文字列から，“PC-6001”だけを抜き出すのです．LEFT$とRIGHT$を組み合わせればできますよ．考えてみてください．

次のようになりますね．

```
? RIGHT$(LEFT$("わたしは PC-6001 です
",12),7)
```

または，

```
? LEFT$(RIGHT$("わたしは PC-6001 です
",11),7)
```

どうです．ちゃんと抜き出せましたね．しかし，わかりにくいですね．実は，N60-BASICには，文字列のまん中を抜き出す命令があるのです．それが，MID$関数です．



これを使えば，

```
? MID$("わたしは PC-6001 です",6,7)
```

というふうに，ずっと簡単になりましたね．

では，これを使ってちょっとしたプログラムを作ってみましょう．次のがそれです．

```
100 INPUT "ぶん";S$
110 INPUT "ことば";W$
120 L=LEN(W$)
130 FOR I=1 TO LEN(S$)
140 IF W$=MID$(S$,I,L) THEN 180
150 NEXT I
160 PRINT "そんな ことば は ぶんの なかに あ
りません。"
170 END
180 PRINT "それは ぶん の";I;" もじめ か
ら にあります。"
190 GOTO 100
```

実行すると，入力を求めてきますから，まずは，

```
ごきげん いかが
```

と入れてください．また，入力を求めてきましたので，今度は，

```
いか
```

と入れましょう．すると，

```
それは ぶん の 8 もじめ から にあります。
```

と表示されましたね．そして，再び入力を求めてきましたので，さっきの文をそのまま残しておくために，RETURN キーだけをおしましょう．それから，次の入力を今度は“たこ”としてみました．すると，

```
そんな ことば は ぶんの なかに ありません。
```

と表示されてプログラムが終わってしまいました．

では，プログラムの説明をしましょう．重要なのは MID$関数によって，最初に入力した文の中から，次に入力された単語を探しているところです．I で示されている第2引数は，S$という文字列の，先頭からの位置を指定しています．Lはその位置からの文字列の長さを示しています．プログラム中では単語の長さですね．そのようにして文の中から抜き出した文字列を，単語と同じかどうか順に比較しているわけです．

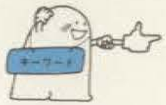
ではこの MID$関数についていろいろと探ってみることにしましょう．

```
? MID$("ABCDE",2,3)
```

とやると，当然ながら，

```
BCD
```

となりますね．では第2引数を0にしてみたらどうなるでしょうか．そうすると，




```
?FC Error
```

となってしまいました．第2引数は，抜き出そうとする文字列の最初の位置を指定するものでしたね．ということは，0の位置というのはありえませんから，エラーになるわけです．同様にして負の数もエラーになります．

次に，第1引数 "ABCDE" よりも長い値を入れてやったらどうなるでしょう．またエラーになりそうですね．

```
? MID$("ABCDE",6,3)
```

ところが，今度はエラーになりませんでした．でも，何も表示されていません．おかしいですね．確かに，6の位置には文字はありません．これはなぜかというと，この関数の場合，第2引数が第1引数の文字列の長さよりも大きい値のときには，ヌルストリングを与えるという約束になっているからです．

このように，論理的にはおかしくてもエラーにはならないものが，N60-BASICにはいくつかありますが，それらは全て，約束ごととして決められていることなので，深く考えないでください．

では，今度は思い切って省略してみましょう．案外できるかもしれませんよ．




```
? MID$("ABCDE",,3)
?SN Error
```

となってしまいました．

今度は第3引数の方をいじってみましょう．これもまず0を入れてみることにしましょう．

```
? MID$("ABCDE",2,0)
```

何も表示されませんでした．それもそのはず，この引数は，取り出す文字列の長さの指定ですから，0なら当然ヌルストリングが与えられるわけです．

では，次に負の数を入れてみましょう．なんとなくできそうな感じがしますけれども，どうでしょう．

```
? MID$("ABCDE",4,-2)
?FC Error
```



残念！ 4つ目から－2文字抜き出して“CD”が出るかと思いましたが，エラーになってしまいましたね．

では今度は，第2引数で示される位置から後ろにある文字数よりも大きい数を入れてみたらどうなるでしょう．

```
? MID$("ABCDE",4,3)
```

この場合はエラーではなく，残りの全てが表示されましたね．では今度は省略してみましょう．まずはカンマを残しておきます．

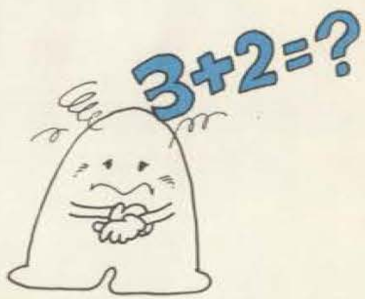
```
? MID$("ABCDE",3,)
?SN Error
```



さっきのような文法上のエラーになりました．では，カンマもとってしまいましょう．

```
? MID$("ABCDE",3)
```

すると今度は，3番目から以降が全て表示されましたね．一見，RIGHT$関数と同じように見えますが，よく見ると違いますね．わからない人は考えてみてください．



では，この節で出てきた文字列関数を使って，次のサンプル・プログラムを作ってみました．これは，“HIT & BLOW”と呼ばれている数あてゲームです．

まず，PCが4桁の数を考えます．この数は1から9までの数字の組み合せで，同じ数字は1度しか使えません．これを“秘密の数”とします．

それをきみが当てるのです．これと思った4桁の数を入 力すると，その1つ1つの数字が秘密の数にあるかないかを調べて，その数を“Blow”，そして位置まで合っている数を“Hit”として教えてもらえます．

たとえば，秘密の数が“6891”だったときに，“2816”と入 力すると，6・8・1は秘密の数にありますから Blow は3，そして8は位置もあっていますから Hit は1となります．

正解のときは，もちろん Hit が4，Blow も4ですね．さて，ゲームの説明は以上です．プログラムを紹 介しますが，力だめしのつもりできみも見ないで作ってみてください．では，がんばって！



```
100 REM *** Hit & Blow ***
110 CONSOLE 0,16:CLS
120 PRINT "かす゛を かんか゛えています。"
130 N$="123456789":S$=""
140 FOR I=1 TO 4
150 N=INT(RND(1)*9)+1
160 A$=MID$(N$,N,1)
170 IF A$="*" THEN 150
180 S$=S$+A$
190 N$=LEFT$(N$,N-1)+"*"+RIGHT$(
N$,9-N)
200 NEXT I
300 INPUT "かす゛を あててくた゛さい。";A$
310 IF LEN(A$)<>4 THEN PRINT "かす
゛は 4 けたて゛す。":GOTO 300
320 H=0:B=0
330 FOR I=1 TO 4
340 FOR J=1 TO 4
350 IF MID$(S$,I,1)=MID$(A$,J,1)
 THEN B=B+1
360 NEXT J
370 IF MID$(S$,I,1)=MID$(A$,I,1)
 THEN H=H+1
380 NEXT I
390 PRINT "Hit =";H,"Blow =";B
400 IF H<4 THEN 300
410 PRINT "おおあたり !!":END
```

2－2　文字←→数値？

初級編のINKEY$を使ったプログラムには，1つもの足りない点がありませんでしたか．それは，カーソルキーや，RETURN キーの判定ができなかったことです．

これらの特殊キーは，そのままでは，どうやっても表わすことができなかったからです．けれども，INKEY$に入力されていないわけではありません．それは，次のような簡単なプログラムを使って証明することができます．

```
100 A$=INKEY$:IF A$="" THEN 100
110 PRINT "おわり":END
```

このプログラムは，INKEY $に何か入力があると，終了するものです．これによると，カーソル移動キーと RETURN キーだけでなく，CLR·HOME キー，INS キー，DEL キー，そして TAB キーも入力されていることがわかります．

ところで，こういう特殊キーは，一体どのようにして INKEY $に読みこまれているのでしょう．実は，これらを含めた全てのキー（ただし，CTRL キー，GRAPH キー，STOP キー，かなキー，そして SHIFT キーは例外です）は，キャラクタ・コードという数値に変換されているのです．

といっても，INKEY $はその名に“$”が付いていることからもわかるとおり，キー入力を文字として与えるものですから，次のようにしてキャラクタ・コードを読みとろうとしても無駄ですよ．

```
100 A$=INKEY$:IF A$="" THEN 100
110 PRINT A$:END
```

どうなるかは，RUN させてみればわかります．つまり，キャラクタ・コードを読みとるには，文字を変換しなければならないのです．そのために N60-BASIC には，ASC 関数が用意されています．



それでは，これをさっきのプログラムに付け加えて，押したキーのキャラクタ・コードを表示さ

せるプログラムにしてみましょう.

```
100 A$=INKEY$:IF A$="" THEN 100
110 PRINT ASC(A$):GOTO 100
```



これを実行させて，いろいろなキーのキャラクタ・コードを調べていきましょう．そうすると，RETURN キーが13，カーソル移動キーは右が28，左が29，上が30，下が31であることがわかりますね．

また，一見同じように見えるキャラクタの違いもわかります．たとえば，マイナス記号とカナ文字の“ー”は，見ただけでは全く同じですが，やってみるとわかるとおり，かなの“ー”はマイナスの代わりにはなりません．これは初級編でやりましたね．そこで，これらのキャラクタ・コードを調べてみましょう．マイナス記号は45，かなの“ー”は176です．英字・数字・特殊文字・かな文字を調べてみましょう．

“日”はいくつでしょう．GRAPH キーを押しながら1を押せばいいですね．あれっ！ 1つの文字しか入力していないのに，2つのコードが出てきました．他も試してみましょう．GRAPH キーを使って入力するものは，どれも2つずつキャラクタ・コードが出ますね．そして，最初のコードはどれも20です．

実は，グラフィック文字の場合は特別に，グラフィック・シフト・コードが付いているのです．1文字なのに2つのコードがあるなんて変わってますね．では試しにグラフィック文字の長さをみてみましょう．1度プログラムをストップして，試してみましょう．




```
? LEN("日")
```

やはり2となりましたね．まとめると，グラフィック文字は，20というグラフィック・シフト・コードと，それぞれのキャラクタ・コードによってできているわけです．そして，表示は1文字でも，2文字分の長さであるわけです．

このようにして，全ての文字のキャラクタ・コードがわかったところで，RETURN キーを押したときだけ実行を終えるプログラムを作ってみましょう．これはわりと応用のできるプログラムです．

```
100 A$=INKEY$:IF A$="" THEN 100
110 IF ASC(A$)<>13 THEN 100
120 PRINT "おわり":END
```

ところでなにげなくついている100行後半ですが，ヌルストリングの場合，キャラクタ・コードはどうなるのでしょうか．調べてみましょう．

```
? ASC("")
?FC Error
Ok
```



となりました．それもそのはず，ヌルストリングにはコードが無いのです．

えっ？　なんですか．　文字をキャラクタ・コードに変える関数があるなら，キャラクタ・コードを文字に変える関数もあるんだろう，ですって？　そうですね．いいところに気がつきましたね．
実は，そのとおり，CHR $関数というものがあるのです．これを使えば，文字として表示できない　カーソル移動キーやRETURNキーを表したり，さらには，ビープ音なども出すことができます．つまり，数値でもってキーボード上のあらゆるキーを表すことができるのです．
また，INKEY $といっしょに使って，押されたキーの判定をすることもできるでしょう．たとえば，さっきのASC関数を使って，RETURNキーを押したときだけ終了するようなプログラムを作りましたが，それをCHR $関数で書き換えると，次のようになりますね．



```
100 A$=INKEY$:IF A$="" THEN 100
110 IF A$<>CHR$(13) THEN 100
120 PRINT "おわり":END
```

その他，グラフィック文字ももちろん表示できます．では，このCHR $関数を使ってN60-BASICの持っている全てのキャラクタを表示させるプログラムを作ってみましょう．

```
100 FOR I=0 TO 255
110 PRINT CHR$(I);
120 NEXT I
130 END
```

“ピッ”という例のエラー音がして，CLS 命令を入れていないのに画面がかってにクリアして，そして画面にいろいろな文字が出ましたね（本当にエラーだった人は，なおしてくださいよ）．

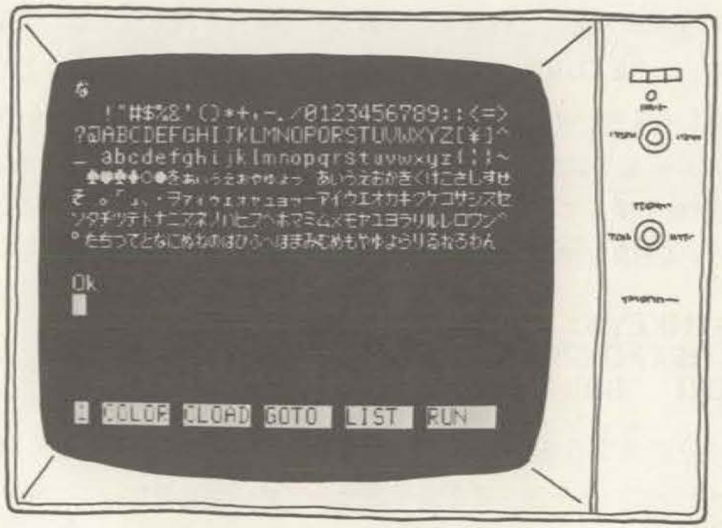
さて，このエラー音が出たり，画面がクリアされたりするのは，コントロール・コードのいたずらです．たとえば，エラー音はコントロールG（コードの7），画面クリアはコントロールL（コードの12）です．もちろん，これはキーボードからでも入力できますよ．試しておいてください．

次にホームポジションの“な”は何かを考えてみましょう．さきほど，グラフィック文字は，コード20に続くコードで表されていることを発見しましたね．では，次のようにキーインしてみてください．



```
? CHR$(20);CHR$(21),CHR$(21)
```

これで，“な”の秘密はわかりましたね．



それでは，じっくりと画面に表示された文字を見てみましょう．おやっ，とまず気がつくことは，4つのトランプマークと白丸・黒丸のグラフィック文字があることです．というのは，グラフィッ

ク文字は，コード20を付けないと出ないはずだからですが……．では，次のように打ちこんでみてください．

```
? CHR$(128),CHR$(20);CHR$(176)
```

どうなりましたか．スペードが２つ表示されましたね．グラフィック文字のうちでもこの６字だけは例外で，普通の文字としてのコードももっているのです．当然，長さも１ですよ．

次に気がつくことは，小文字のｚの次の４つの文字です．これは，キーボードには無い文字ですね．でも，あることがわかってしまうと使いたくなるのが人間です．きみもそうでしょ！

では，この文字をプログラムで使う方法をこっそりお教えましょう．PCは画面上にある文字は，別にキーで入力されたものでなくても，ちゃんと文字として扱ってくれます．そこで，たとえば"～"をPRINT文中に入れたいのなら，まず，次のようにします．

```
? chr$(126)
~
```



次の行の１文字目に"～"が出ましたね．そうしておいて，カーソルをその上に移動し，次にINSキーを押してから，その前に来るべき文字をキーインしていけばいいのです．

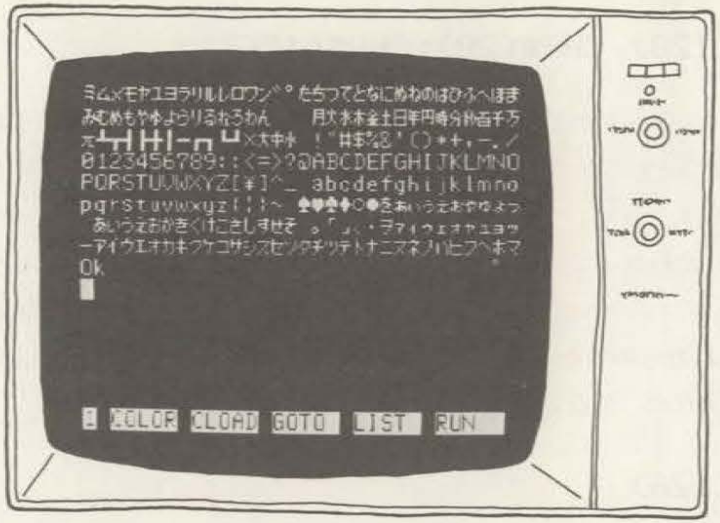
わかりましたか．読んでわからなくても実際にやってみればわかりますよ．"読むより，触って試せ！"ですから……．

話を戻しましょう．次のプログラムを入力，実行してみてください．

```
100 FOR I=0 TO 255
110 PRINT CHR$(20);CHR$(I);
120 NEXT I
130 END
```

これは，さきほどのプログラムにちょっと手を加えたものですね．どこが違うかはご覧のとおり，普通にそのままコードを文字に変換して表示する前に，コード20を出しているだけです．しかし，今度はエラー音を出したり，自動的に画面をクリアしたりしません．

そして，グラフィック文字を含めて，PCの持っている全ての文字が表示されていますね．



2－3　数値←→文字列？？

ところで，この章の第1節で文字列関数が4つでてきましたが，使い方がわかると，“これは便利だ！”と思ったことでしょう．こんなに便利な関数を文字列だけのものにしておくなんて，もったいないと思いませんか．数値にも，あれらの関数が使えたらいいのになあ，とがっかりするのは早いですよ．



そうです．N60-BASICには，数値を文字列にしてしまう関数があるのです．以後，文字列化した数値を“数字列”と呼ぶことにします．それでは，数値を数字列に直す方法です．それには，STR$関数を使います．

数値の表示は，左づめで表示されることはすでに知っていますね．でも，きれいな表をつくるには，右づめにした方が見やすいでしょう．そこで，STR$関数の使い方といっしょに，数字を右づめに表示するテクニックもお教えしましょう．

では次のプログラムを打ちこんでください．

```
100 N=-1000000
110 N=N/10:N$=RIGHT$("      "+STR
$(N),7)
120 PRINT N,N$
130 IF N<-1 THEN 110
140 N=.1
150 N=N*10:N$=RIGHT$("      "+STR
$(N),7)
160 PRINT N,N$
170 IF N<1E5 THEN 150
180 END
```

実行させますと，比較のために，数値を左づめに普通に表示して，数字列を右づめに表示します．実行が終わったあとの画面は，ダイヤモンド形のようになって見えますね．

プログラムをよく見てください．RIGHT$関数が使われていますね．これによって，数字列を全て7文字に切りそろえているわけです．つまり，右側の数字列群は，7文字の長さの文字列として表示されているわけなのです．

このように，見やすい表示を作るためにも，STR$関数はとても便利なものです．そして，文字列化した数値を，さっきの例のように，文字列関数を使って処理できるわけです．

1度数字列にしてしまった数値は，そのままでは，演算ができなくなります．そこで，数字列をもとの数値に戻す関数もあります．N60-BASIC では，VAL 関数といいます．



たとえば，INPUT 文で，変数を普通の数値変数にしてしまった場合に，文字を入れたりすると，

```
?Redo from start
```

が表字されて，表示が汚くなってしまって，いやな思いをすることがあります．こういう時に，INKEY$と VAL 関数を使って，数値しか受けつけないような入力ルーチンを作っておくと，きれいな入力ができます．

では，以上の機能の他にも，打ち間違いを直せる機能をつけ加えた，実用的な数値入力プログラムを作ってみましょう．



```
100 PRINT "かず を いれてください。 ";
110 GOSUB 200
120 PRINT NUM
130 END
200 N$=""
210 A$=INKEY$:IF A$="" THEN 210
220 IF A$="0" THEN 270
230 IF A$="-" THEN IF N$="" THEN
 270
240 IF A$=CHR$(8) THEN IF N$<>""
 THEN A$=CHR$(29):GOTO 280
250 IF A$=CHR$(13) THEN 290
260 IF VAL(A$)=0 THEN 210
270 PRINT A$;:N$=N$+A$:GOTO 210
280 PRINT A$;" ";A$;:N$=LEFT$(N$
,LEN(N$)-1):GOTO 210
290 PRINT:NUM=VAL(N$):RETURN
```

実行させて，適当な数値を入れてください．数値キー以外は受けつけません．試しに，英文字を押してごらんなさい．ただし，マイナス記号は，最初の1字目に限って入力が許されます．



いくつかの数値を入れたところで，DEL キーを押してみてください．1文字ずつ消えてゆきます．そのようにして間違い訂正ができます．けれども，INPUT 文とは異なり，表示された文まで破壊してしまうことはありません．

数値を入力し終わったら，RETURN キーを押してください．これで数値入力サブルーチンから戻って，入力された数値が表示されます．

どうです．これなら，コンピュータを知らない人に使わせても間違いが起こりませんね．このルーチンを参考にして，そのときに必要なキー以外は入力できなくすることができるでしょう．きみもどんどん使ってください．

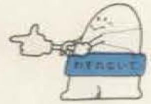
ところで，VAL 関数にはいろいろと決まりがあります．たとえばまず，次のように打ちこんでみてください．

```
? VAL("-123")
```

もちろん答は−123ですが，では今度は？

```
? VAL("3-1")
```

一瞬だまされたでしょう．この数字列“3−1”が数値になると，“2”になるような気がしますね．でも答は，“3”でした．つまり，VAL関数は，文字列の最初のマイナス記号は負の符号として読み取りますが，数字の後のマイナス記号は，文字として読み取ってしまうのです．そして，文字の後の数字はよみとらないという性質があります．



では，次にもう1つやってみてください．

```
? VAL("3.14")
```

答は，もちろん3.14でしたね．VAL関数は小数もよみとることができます．では今度は？

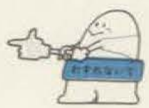
```
? VAL("3.1.4")
```

今度はあまりおどろきませんでしたね．この場合もさっきのマイナス記号と同様に，2つ目以降の小数点は文字として読み取ってしまうわけです．

もう1つおまけです．次のように打ちこんでみてください．

```
? VAL("1e2")
```

ちゃんと“100”になりましたね．英文字の中でもこの“e”だけは特別です．もちろん大文字でも大丈夫ですよ．



このようなVAL関数の性質をよく覚えておいて，間違えないようにしてくださいね．割と勘ちがいしやすいものです．

では，この節でててきた型変換関数のまとめとして，それらを使って，ローマ字をカナ文字に変換するプログラムを作ってみました．

```
100 DIM A(25)
110 FOR I=0 TO 25:READ A(I):NEXT
120 BS$=CHR$(29)+" "+CHR$(29)
130 PRINT "*** ちゅうい じこう ***"
140 PRINT "1. こもじを つかうこと"
150 PRINT "2. 'ン' は '-'"
160 PLAY "O5L64E"
200 PLAY "C"
210 C=177:D=0
220 A$=INKEY$:IF A$="" GOTO 220
230 A=ASC(A$):IF A>96 GOTO 300
240 IF A=32 THEN PRINT " ";
250 IF A=13 THEN PRINT
260 IF A=8  THEN PRINT BS$;
270 IF A=45 THEN PRINT "ン";
280 GOTO 210
300 I=A-97
310 IF A(I)=0 GOTO 200
320 IF A(I)<6 GOTO 380
330 C=A(I)
340 IF C>355 THEN C=C-200:D=2
350 IF C>255 THEN C=C-100:D=1
360 GOTO 220
380 B=A(I)
390 IF C=212 THEN B=INT(B/2)+1
400 PRINT CHR$(C+B-1);
410 IF D=1 THEN PRINT "゛";
420 IF D=2 THEN PRINT "゜";
430 GOTO 210
450 DATA 1,302,0,292,4,0,282
460 DATA 202,2,0,182,0,207,197
470 DATA 5,402,0,215,187,192,3
480 DATA 0,220,0,212,287
```

2－4　文字列の大小関係

第1節では，文字列が数値よりも便利な関数をいくつか持っていることを知りました．第2節，第3節では，数値と文字の間のさまざまな変換方法を知りましたね．では，もう1つ，文字列にできそうなものはなんでしょう．

そうです．文字列の大小比較をまだ試していませんね．文字列も，数値のような比較演算子を使って，大小比較ができるのでしょうか．今までにも等しい場合については，すでにいろいろなところで出てきています．

結論から先にいうと，文字列も大小比較ができます．数値の大小比較は，もちろん充分わかっていると思いますが，では文字列は何を基準にして，大小判定をしたらいいのでしょう．数値には－∞（無限大）から始まって，0を通り，＋∞へつながってゆく，ものさしがあります．

そこで，文字列のものさしとなるものが，キャラクタ・コードなのです．このキャラクタ・コードの大小によって文字の大小を判定するのです．ただし，文字列の比較の場合は，いくら長いものであっても，それらのキャラクタ・コードを合計したりするわけではありません．

文字列の先頭から，1文字ずつキャラクタ・コードによって比較するわけです．そのため，数値の場合とは異なる場合があり，錯覚しやすいので気をつけましょう．



また，英文字はキャラクタ・コードの順序と同じですが，ひらがなやカタカナの場合は，50音順にはなっていません．このことは，CHR$関数の説明のところで全キャラクタを表示したときにわかったことと思います．さらに，グラフィック文字の1部には頭にコード20がついていますので，その点にも注意しなくてはいけません．そのような出力されない文字や，空白も当然意味を持ちます．これらのことをまとめてみますと，次のような例がいくつか考えられます．

```
"A"="A"
"AAA"<"B"
"キュウ">"キュー"
"4-3">"2"      (4-3<2)
"大"<"!"
"ゅ"<"ュ"      ("ゅ">"ユ")
"A"<"a"
```

いちいち，コードで見比べるのは不便ですから，ここは PC におうかがいをたてましょう．次のプログラムで聞いてください．




```
100 INPUT "A$,B$";A$,B$
110 PRINT A$;
120 IF A$>B$ THEN PRINT " > ";
130 IF A$=B$ THEN PRINT " = ";
140 IF A$<B$ THEN PRINT " < ";
150 PRINT B$:PRINT
160 GOTO 100
```

2－5　なんて書いてあるの，これ？

突然ではありますが，まず画面をクリアして，ホームポジションから次のようにキーインしてみてください．



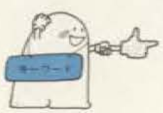

```
? SCREEN (0,0)
```

なにごとも，エラーを恐れていたのでは，新しいことを発見することはできませんよ．さあ，迷わず，RETURN キーを押してください．

どうです．エラーなどにはならずに 63 と表示されたでしょう．かんの鋭いきみなら，ピンときたでしょう．そこで，

```
? CHR$(63)
```

と打ちこむと，思ったとおり“?”が表示されましたね．



そうなんです．初級編の第 10 章で登場したこの SCREEN は実は，1人2役なんです．つまり，1つはモードやページを変えるための命令，そしてもう 1つは画面上のキャラクタ・コードを読み取るための関数として，使えるわけです．

次のプログラムを実行してみてください．

```
100 CONSOLE 0,16,0:CLS
110 FOR V=0 TO 14 STEP 14
120 LOCATE 0,V:PRINT "oooooooooo
oooooooooooooooooooooo";
130 NEXT V
140 FOR V=1 TO 13
150 LOCATE 0,V:PRINT "o";
160 LOCATE 31,V:PRINT "o";
170 NEXT V
200 FOR N=1 TO 50
210 H=INT(RND(1)*30)+1
220 V=INT(RND(1)*13)+1
230 LOCATE H,V:PRINT "*";
240 NEXT N
250 NH=INT(RND(1)*30)+1
260 NV=INT(RND(1)*13)+1
270 OH=NH:OV=NV:X=1:Y=1
300 LOCATE NH,NV:PRINT "●";
310 NH=OH+X:NV=OV+Y
320 IF SCREEN(NH,NV)=32 THEN 400
330 X=INT(RND(1)*3)-1
340 Y=INT(RND(1)*3)-1
350 LOCATE 0,15:PRINT X,Y;:GOTO
310
400 LOCATE OH,OV:PRINT " ";
410 OH=NH:OV=NV:GOTO 300
```

画面の下の方の数字は，“●”の進行方向を示しています．“○”に当たった時はいろいろ考えているように変わっていきますね．

さて，ここで問題となるのは，グラフィック・シフト・コードのあるグラフィック文字です．これをSCREEN関数で読み取るとどうなるでしょう．

画面をクリアしておいて，“日(GRAPH-1)”をキーインして，カーソルをその下に動かしてから，

```
? SCREEN (0,0)
```

とすれば確かめられますね．いくつになりました．そう，7ですね．しかし，7といえばコントロールGのはずです．

実は，グラフィック文字をSCREEN関数で読み取ると，普通は画面に存在するはずのない，コントロール・コードが得られるのです．他のグラフィック文字でも同じようにして調べておいてください．



第3章　さらに便利な命令



第2部もいよいよ第3章になりました．第1章でスクリーンのコントロール方法を知り，第2 章で思いどおりの表示ができるようになりましたね．

そろそろいいかげんに，BASIC の命令が尽きるころだろう，なんて思っていませんか．

残念ながら，まだまだ大事な命令があります．でも，もう少しですから，がんばってください．



3－1　今，何時ですか

きみは，コンピュータがクォーツ時計を持っているのを知っていますか．

クォーツ時計といっても，時刻を知るためにあるわけではありません．コンピュータの中にはたくさんの信号線があり，一定の規則に従って電気が流れています．その規則を守るためには，正確なタイミングが必要です．

そのタイミングの基準として，このクォーツ時計が使われています．そして，これを“クロック”と呼んでいます．きみの PC-6001 もコンピュータですから，当然クロックを持っています．



ところで，ゲームに時間切れを付けたいときがありますね．そういうときに，わざわざストップウォッチを見ながらやっていたのでは，おもしろくありません．PC の時計をちょっと借りられれば便利ですね．

N60-BASIC には，ちゃんと PC の時計を借りる命令があります．ただし，この時計は，きみが日頃使っているような置時計や腕時計のように，今の時刻を教えてくれるわけではありません．

では何かというと，実は PC-6001 のクロックを数える，“カウンタ”というものなのです．そして，このカウンタは PC に電源を入れたとたんにクロックを数え始めます．



おっと，説明ばかり長くなってしまいました．実際の使い方を忘れていました．N60-BASIC では，



“TIME” という変数の中にカウンタの値が入っています．ですから，この TIME を読み取れば良いわけです．

では，どのくらいの速さでカウントされているのかをみてみましょう．それは次のような簡単なプログラムでできます．

```
100 CLS
110 LOCATE 0,0:PRINT TIME
120 GOTO 110
```

実行してください．どうですか，とても速く数が増えています．下2ケタまでは，とても目で見ることなどできないくらいです．

これを使って，プログラムの実行速度を見てみましょう．

```
100 PRINT TIME
110 PRINT TIME
120 PRINT TIME
130 END
```

3つの数が出ましたが，わずかに40～60くらいしか違いませんね．コンピュータがいかに速く動作しているのかがわかりますね．

ところで，この TIME の値をプログラムの実行開始とともに，0にセットできれば便利だと思いませんか．TIME は，一見変数のようですから，代入もできるはずです．では何か代入してみましょう．

```
TIME=0
```

と打ちこむと，



```
?SN Error
```

となってしまいました．ということは，TIME は，CSRLIN と同様に特殊変数なのです．そうなると，0にセットすることはあきらめるしかありません．では，どのように使ったらよいのでしょうか．

次のプログラムをみてください.

```
100 ST=TIME:CLS
110 LOCATE 0,0:PRINT TIME-ST
120 IF TIME-ST<1000 THEN 110
130 END
```

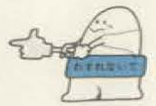
このような，プログラムの最初に普通の変数へ代入してしまって，後は差を使えばいいのです．では実行してみてください．止まったときの数字をよく見てください．普通は，980 前後ですが，たまに， 1000 なんていうのがありますね．これは表示したときと，IF 文で比較したときの時間差によるものです．他のプログラムに応用する際には，充分注意してください．

では，TIME の値から，分秒表示のデジタル時計に変換するプログラムを作ってみましょう．

```
100 CONSOLE 0,16:CLS
110 ST=TIME
120 NT=TIME
130 S=INT((NT-ST)/976)
140 IF S>=60 THEN M=M+1:S=0:ST=T
IME
150 IF M>=60 THEN H=H+1:M=0:ST=T
IME
160 LOCATE 0,0
170 PRINT "TIME:";RIGHT$(STR$(H)
,2);
180 PRINT ":";RIGHT$(STR$(M),2);
190 PRINT ":";RIGHT$(STR$(S),2);
200 GOTO 120
```

どうです．ちゃんとした時計になりましたね．これに入力ルーチンを付けて，時刻合わせができるようにするのはきみの役目ですよ．

ところで，TIME を“976”で割って秒に変換しているのがわかりますか．これが，TIME の“補正値”です．覚えておいてください．



3－2　1だったら，2だったら，3だったら

ある変数の値が，1ならこれをして，2だったらあれをして，3では……，などということがよくあります．たとえば，1つのプログラムに違うゲームが，たくさん入っていて実行すると，どれをやるか聞いてきてその答をもとに，希望のゲームに移る場合などがそうです．

そんなとき，きみならどうします．IF 文で1つ1つジャンプさせますか．2つや3つに分かれるのならそれもいいでしょう．しかし，5つや10に分かれるとなるとちょっとめんどうですね．

BASIC には，そのための命令が用意されているのです．それが，ON-GOTO 文なのです．



ON-GOTO 文では，ON の次の値によって，GOTO の次にカンマで区切って並べられている複数の行番号の何番目にジャンプするかを決められる命令です．たとえば，次のようにすると，

```
100 ON A GOTO 200,300,400,500
```

Aが，1のとき200へ，2のとき300へ，3のとき400へ，4のとき500へ，となるわけです．では次のプログラムを実行してください．




```
100 INPUT "A=";A
110 ON A GOTO 200,300,400,500
200 PRINT "A は 1 です。":GOTO 100
300 PRINT "A ﾊ 2 ﾃﾞｽ。":GOTO 100
400 PRINT "A = 3  !!!":GOTO 100
500 PRINT "A is 4 . ?":GOTO 100
```

まず，1を入れてみましょう．

```
A は 1 です。
```

と表示されましたね．2，3や4も試してみてください．1～4は，それぞれ飛び先があるわけですが，飛び先の無い5以上の数ではどうなるでしょう．試してみましょう．どうなりましたか．1のときと同じですね．これはどういうことなのでしょうか．次の1行を付け加えてみましょう．

```
120 PRINT "5いじょうです。":GOTO 100
```

さて，今度はどうでしょう．5を入力してみてください．次のように表示されましたね．

```
5いじょうです。
```

では，もっと大きな数だったらどうなるでしょう．きみ自身の手で確かめておいてください．答は，256以上ではエラーになってしまいます．

このように，行番号の個数より大きな値だった場合は，255以下ならON-GOTO文を無視して次の文を実行するのです．エラーにならない点に注意してください．



0や負の数はどうでしょう．まず，0では，これは5のときと同じですね．次に負の数ですが，試しに−1を入力してください．

```
?FC Error in 110
```



でエラーになってしまいましたね．負の数ではすべてエラーです．他の数でも試しておいてください．

それではさらに，小数ではどうでしょう．これも，きみの手で確かめておいてください．

ところで，ON-GOTO文があるのなら，ON-GOSUB文もあるとは思いませんか．もちろんあります．使い方は，ON-GOTOと同じですが，こちらの場合はサブルーチンから戻ってきて，次の文を実行することを忘れないでくださいね．



さて，このようなON-GOTO文やON-GOSUB文を“多分岐命令”といいますが，ONの後の値が1や2のような小さな数だけでは不便と思うでしょう．



しかし，ちょっと工夫すれば，いろいろな条件で分岐することができるのです．たとえば，さきほどのプログラムを次のように変えると，

```
100 INPUT "A=";A$
110 ON ASC(A$)-96 GOTO 200,300,4
00,500
```

プログラム続く



```
120 PRINT "5いじょうです。":GOTO 100
200 PRINT "A は 1 です。":GOTO 100
300 PRINT "A ハ 2 デス。":GOTO 100
400 PRINT "A = 3  !!!":GOTO 100
500 PRINT "A is 4 . ?":GOTO 100
```

今度は，数の1や2ではなくて，文字のaやbで飛び先が決められるようにすることもできるのです．きみもいろいろと工夫して，使ってください．

そうそう，もう1つ大事なことを忘れていました．もし飛び先の行番号の行がなかったらどうなるでしょう．さっきのプログラムの1行を次のように変えてやってみてください．

```
110 ON ASC(A$)-96 GOTO 200,300,4
00,5000
```

aやbでは大丈夫ですが，dでは，



```
?UL Error in 110
```

となりましたね．これは，5000の行が無いからです．次のような場合もこのエラーです．

```
ON A GOTO
```

後でプログラムを付け加えよう，なんて思って忘れてしまったときによく出るエラーですが，あわてないようにしてくださいね．

3－3　IF文，大活躍！

IF文による条件判断は，初級編の第7章でやりましたね．きみはしっかり覚えていますか．忘れていたら，もう1度戻ってしっかり身につけておいてください．



さて，たとえば，“A=3”という条件をきみが判断するなら，Aが3なら“はい”つまり“真”，Aが3じゃなかったら“いいえ”つまり“偽”ということばで表しますね．ところが，コンピュータ

の内部には，こんなことばはありません．

それでは，どのようにして判断するのでしょうか．ここで，ちょっとおもしろいことをしてみましょう．まず，次のようにします．

```
a=1
```

これは，単に変数 a に1を代入しているだけですから，なんてことはありませんね．では，次に，




```
? a=-1
```

と打ちこんでください．おかしいと思うでしょうが，とにかく打ちこんでください．どうなりましたか．

```
0
```

と表示されましたね．次に，

```
? a=1
```

では，どうです．

```
-1
```

となりましたね．“=”の次をいろいろな数にしてやってみてください．

実はこれが，N-60BASIC の真と偽なのです．では，次のプログラムを実行してみてください．

```
100 FOR I=-2 TO 2
110 L$="ぎ":PRINT I,
120 IF I THEN L$="しん"
130 PRINT L$
140 NEXT I:END
```

結果は次のようになるはずです.

```
 -2          しん
 -1          しん
  0          ぎ
  1          しん
  2          しん
```

つまり,

```
IF I THEN
```

というのは,

```
IF I<>0 THEN
```

と同じだということです.



このように, N60-BASICでは, 式の値が"0"のとき"偽", それ以外のとき"真"で, 式の値としては"−1"になるのです. そして, この0と−1を"論理値"といいます.



さて,ちょっと難しい話で始まってしまいましたが,これを使ってIF文でたくさんの条件をまとめて扱おうというのです. そこで登場するのが, AND, OR, NOTという"論理演算子"です.

まず, ANDは"そして"と読めばわかりいいでしょう. たとえば,

```
IF (A=1) AND (B>5) THEN
```

とすれば, "Aが1", そして"B>5"のときだけ, THEN以下の文が実行されます. これは条件式が2つですが, さらにANDを使って3つでも4つでもまとめて判断できます.

では, 具体的にANDの働きを見てみましょう. 次のプログラムを実行してみてください.

```
100 FOR X=-1 TO 0
110 FOR Y=-1 TO 0
120 PRINT X;" AND ";Y;" = ";
```

プログラム続く



```
130 PRINT X AND Y
140 NEXT Y,X
150 END
```

```
-1 AND -1 = -1
-1 AND  0 =  0
 0 AND -1 =  0
 0 AND  0 =  0
```

このように，両方が真のときだけ真になります．

次のORは，“または”と読みましょう．たとえば，



```
IF (A=18) OR (A=22) THEN
```

とやると，“Aが18”または“Aが22”のときだけ，THEN以下の文を実行します．ただし，これを，

```
IF A=(18 OR 22) THEN
```

として，“Aが18または22”とはできませんよ．

ANDのときと同じように，ORの働きを見てみましょう．

```
100 FOR X=-1 TO 0
110 FOR Y=-1 TO 0
120 PRINT X;" OR ";Y;" = ";
130 PRINT X OR Y
140 NEXT Y,X
150 END
```

```
-1 OR -1 = -1
-1 OR  0 = -1
 0 OR -1 = -1
 0 OR  0 =  0
```

どうです．ANDとは逆に両方が偽のときだけ偽になりましたね．
よく，ANDとORを間違えることがありますが，たとえば今の例で，

```
IF (A=18) AND (A=22) THEN
```

としたりすると，“Aが18”そして“Aが22”なんていうことは絶対ありませんから，いつまで待ってもTHEN以下が実行されなくなってしまいますから，充分気をつけてください．



それでは，最後のNOTです．英語を習ったことがあれば，おおかたの見当はつくでしょう．そうです．否定ですから，“ではない”と覚えてください．たとえば，

```
IF NOT(A=8) THEN
```

とすれば，“Aが8ではない”ときだけTHEN以下を実行します．このNOTは，他のANDやORと異なり，条件式を右側に1つだけしか書けません．ちょうど符号の“－”と同じような使い方ですね．

では，これも確かめておきましょう．

```
100 FOR X=-1 TO 0
110 PRINT "NOT ";X;" = ";
120 PRINT NOT X
130 NEXT X
140 END

NOT -1 =  0
NOT  0 = -1
```

さて，今までは1つ1つやってきましたが，これを組み合わせることもできます．たとえば，

```
IF (A=0) OR ((B<>1) AND (C>9)) T
HEN X=0:Y=1
```

では，どんなときにTHEN以下が実行されるでしょう．考えてみてください．

この場合は，カッコ優先ですから，“Bが1ではなく”そして“Cが9より大きい”かまたは“Aが0”のときですね．

どうです．IF文に大活躍させる方法はわかりましたか．では，第2部の第1章で作った，“＊”を動かすプログラムにちょっと手を加えて，簡単なゲームを作ってみましょう．

このゲームは，画面に散らばっている20個のポイントを，カーソル移動キーで動く“＊”で消していき，その時間を競うものです．




```
100 SCREEN 1,1,1:CLS:SCREEN 1,2,
2:CLS
110 PRINT "ポイント をかいています。":P=20
120 H=INT(RND(1)*32)
130 V=INT(RND(1)*15)
140 IF H=0 AND V=0 THEN 120
150 SCREEN 1,1,2
160 IF SCREEN(H,V)=132 THEN 120
170 LOCATE H,V:PRINT "o";:P=P-1
180 SCREEN 1,2,2:PRINT P;" ";
190 IF P<>0 THEN 120
200 PRINT:PRINT "ヨーイ ドン !":P=20
:TS=TIME
210 SCREEN 1,1,1:H=0:V=0
220 LOCATE 0,15:PRINT "のこり の ポイ
ント ";
230 LOCATE H,V:PRINT "*";
240 IF P=0 THEN 340
250 A$=INKEY$:IF A$="" THEN 250
260 LOCATE H,V:PRINT " ";
270 IF A$=CHR$(28) AND H<31 THEN
 H=H+1
280 IF A$=CHR$(29) AND H>0 THEN
H=H-1
290 IF A$=CHR$(30) AND V>0 THEN
V=V-1
300 IF A$=CHR$(31) AND V<14 THEN
 V=V+1
```



```
310 IF SCREEN (H,V)=132 THEN P=P
-1
320 LOCATE 12,15:PRINT P;" ";
330 GOTO 230
340 TE=TIME:CONSOLE 0,16
350 LOCATE 0,0:PRINT "ゴール イン !"

360 PRINT "あなた の タイム は ";
370 PRINT INT((TE-TS)/976);" 秒 で
した。 "
380 END
```

この章で現れたいろいろな命令を使ってみました．そのせいで，多少不自然な部分がありますが，よくみてどんどん改良して，立派なゲームに仕上げてください．

さて，話は戻りますが，条件式は必ず "0" か "−1" の論理値に変換されましたね．これをそのまま数値として使うこともできるはずです．たとえば，

```
C=C+(A=0)*(B=0)
```

としたらどうなるでしょう．Aが0のときは，左のカッコの値は−1です．そして，Bも0のとき右のカッコの値も−1ですから．

```
C=C+(-1)*(-1)
```

で，Cが1つ増えることになります．AかBのどちらかが0ではないとき，そのカッコの値は0になりますから，Cはそのままです．IF文で書くならば，

```
IF (A=0) AND (B=0) THEN C=C+1
```

となるわけです．

長さをみてもわかりますが，この例のような場合は，IF文を使わないほうが実行速度が上がります．

第4章　ジョイスティックと，プリンタと



さて，ここまで読み進んできたきみなら，もうどんなことでもできるはずです．もう，どんなふうにプログラムを作るのかとか，1つ1つの命令を根ほり葉ほり探ってみるようなことはしません．というのも，ある新しい命令を見たらきみはどのようにすれば試せるか，そしてどのようなプログラムに使えるのかをすっかり身につけているはずだからです．

もう残るは2つの章だけですから，がんばってください．

4－1　ジョイスティックが欲しい！

きみは，ゲームセンターへ行ったことがありますか．行ったことがない人でも，あの“インベーダーゲーム”は知っているでしょう．

あのゲームもコンピュータによるものですが，なぜブームになったのかというと，自分のミサイルの砲台がレバー1つで左へ，右へ，思いどおりに移動できたことと，自分の思ったときに押しボタンでミサイルを発射できたからです．

つまり，中でコンピュータがけんめいに仕事をしていることを，全く知らなくてもゲームのやり方が機械を見ただけで，簡単にわかるからです．

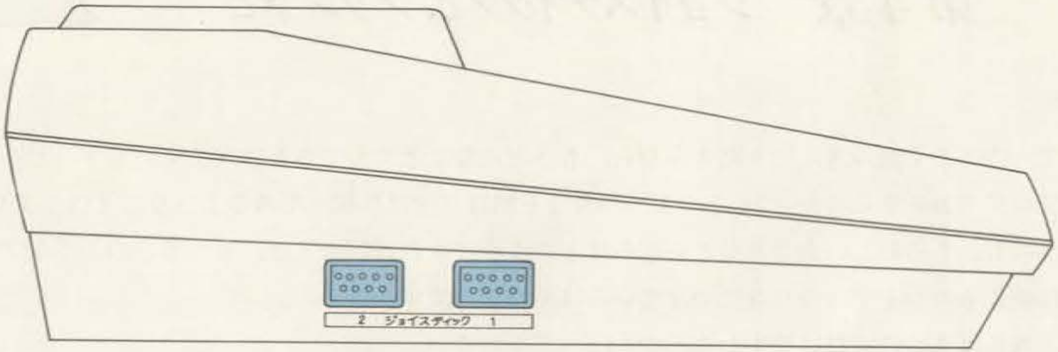
そうです．ポイントはレバーとボタンだったのです．そして，きみのPCにもこれが付けられるのです．



それが，“ジョイスティック”です．これは，四角の薄い箱の中心に1つのボタンと1本の棒が付けられた形をしています．きみがPCを買った店に行けばあるはずです．

そして，この棒は四方八方へ倒すことができるようになっています．どうです．レバーは2方向へしか倒せませんから，この方がずっとおもしろいことができるはずです．

このジョイスティックを，PCの左側面にあるソケットにつなぐことができます．ソケットは2つありますから，2つのジョイスティックをつなぐことができますね．ところが，N 60-BASICでは，3つのジョイスティックを使うことができるのです．



では，もう1つのジョイスティックはどこにつなぐのでしょう．実は，もうPCに付いているのです．



それは，キーボード右端のカーソル移動キー，そしてスペースバーです．つまり，カーソル移動キーが棒の代わり，スペースバーがボタンの代わりになるのです．

それでは，次のプログラムを実行してください．

```
100 CLS
110 FOR I=0 TO 2
120 LOCATE 0,I*2
130 PRINT STICK(I);
140 LOCATE 15,I*2
150 PRINT STRIG(I);
160 NEXT I:GOTO 110
```

そして，カーソル移動キーやスペースバーをいろいろ押してみてください．もちろん，ジョイスティックを持っている人は，つないでやってみてください．



さて，プログラム中の2つの新しい関数，STICK (x) とSTRIG (x) が，このジョイスティックの状態を調べるものであることは，いうまでもありません．いろいろと試しておいてくださいね．

ただし，1つだけ注意しておきます．それは，カーソル移動キーをジョイスティックの代わりに使うときです．1つ1つのキー押しているときは，STICK（0）の値は必ず奇数となっていますね．では，2つずつキーを押してみてください．偶数の値も得られましたね．

このように，STICK 関数の値は，0から8までです．

4－2　画面の表示を保存する

PC がいかに速く計算しても，その結果を人間が画面を見ながら，紙に写していたのではなんにもなりません．

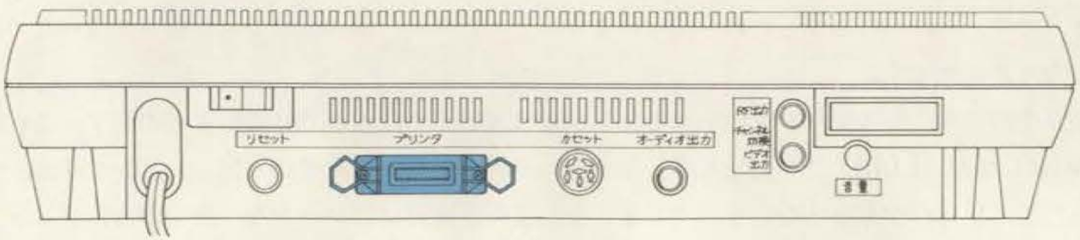
また，きみが普通にプログラミングしたならば，とうていそのリストは画面内に収まりきれないでしょう．かといって，これもいちいちきみが紙に写していたのではたまりません．

この紙に写すという作業を自動的にやってくれる装置，それが“プリンタ”なのです．プリンタはやや高い機械かもしれませんが，あるととても便利なものです．



この節では，プリンタに関するN 60-BASIC の命令を説明します．プリンタを持っている人はもちろん，PC を充分に使いこなそうと思っている人はぜひ読んでください．

まず，しなければならないのは，PC とプリンタをつなぐことです．これには，“プリンタケーブル”という専用の電線が必要で，一方をプリンタに，もう一方は PC の背中にあるコネクタにつなぎます．プリンタの準備は，プリンタの説明書に書いてあるようにしてください．



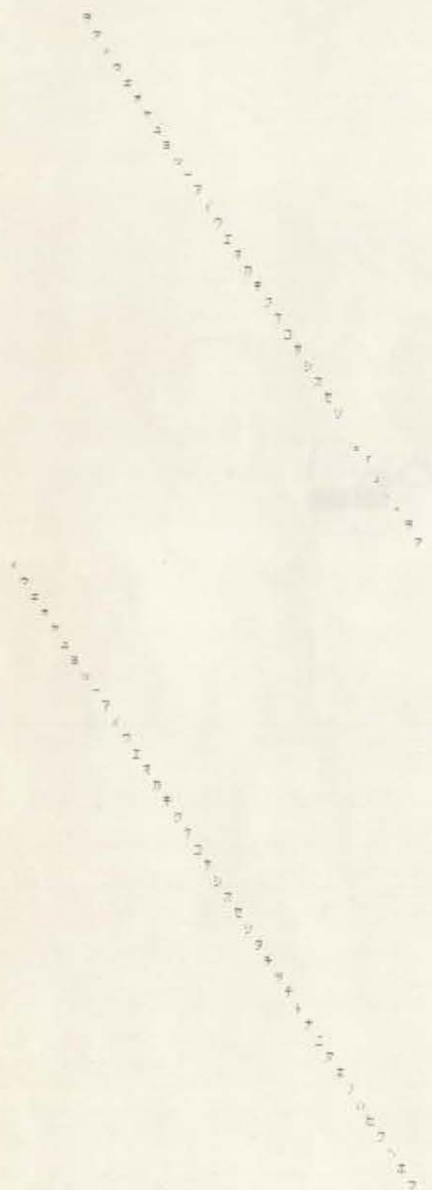
さて，プリンタに関する命令は，画面に表示する命令に似ているだろうということはなんとなくわかるでしょう．実は，画面に対する命令に“L”を付けたものなのです．

まず，プログラムのリストをプリンタにとる命令は，LLIST ということになります．そして，部分的なリストをとるのは，LIST 命令を全く同じようにすればよいのです．

プリンタをつながないで，この命令を使うと PC は無いはずのプリンタを探しにいってしまいますから，STOP キーで探すのをやめさせてください．

次のリストが，PC-6001 専用のプリンタ，PC-6021 でとったものです．

```
100 FOR I=128 TO 200 STEP 40
110 FOR J=0 TO 39
120 PRINT TAB(J);CHR$(I+J)
130 NEXT J,I
```



このように，プリンタから出されたものを，"プリントアウト"といいます．では，このプログラムを PC で実行してみてください．どうなりましたか．ちょっとおかしな感じですね．

実は，このプログラムは，プリンタのためのものなのです．プリンタに出すようにすると美しくできるのです．

そこで，プリンタに好きな文字や数字を出すにはどうしたらいいのでしょう．これも簡単で，画面に表示するための PRINT に L を付けて LPRINT にすればいいのです．

では，さっきのプログラムの PRINT を LPRINT に代えて実行してみましょう．その結果は左のようになります．

どうです．きれいに出ましたね．でも，よくみてください．画面にはひらがなも出ていたのに，みんなカタカナですよ．

そうなのです．実はプリンタに出すとひらがなはカタカナに変換されてしまうのです．これは，LPRINT でも，LLIST でも同じです．しかし，カタカナだけではやはり読みにくいですね．なんとかして，ひらがなも出せないものでしょうか．また，画面にはひらがなが出るのに，プリンタには出ないのはどうしてなのでしょうか．

PC-6001 は簡単に画面にひらがなを表示していますが，実はこれは大変なことなのです．その証拠にちょっと前までのパーソナル・コンピュータでは，PC-6001 よりもずっと値段の高いものでも，

ひらがなは表示できなかったのです．

さて，プリンタに文字を出す場合，PCはプリンタにキャラクタ・コードを送っています．プリンタは，そのキャラクタ・コードに応じて文字を出しますが，プリンタはひらがなの形を知らないのです．そのため，PCは送ろうとしたコードがひらがなだったら，自動的にカタカナの同じ字のコードに変換しているのです．なにも出ないよりは，せめてカタカナででも出てくれたほうがありがたいでしょう．

では，プリンタには絶対にひらがなは出せないのでしょうか．それでは，画面を手で紙に写したほうがいいと，きみなら思うかもしれませんね．

プリンタに，キャラクタ・コードを送るのではなくて，文字の形をデータとして送ればいいと思いませんか．グラフィックスのところで，モード3やモード4は文字を点で描いていたのを思い出してください．

N60-BASICには，ちゃんとそのための命令も用意されています．次のプリントアウトがその例です．

```
      わたしは、 PC-6001 です。
  NEC うまれの パーソナル・コンピュータ です。

 これが、PC-6021:サーマル ほうしきの プリンタ に
よる しゅつりょく サンプル です。

 PC-6021 があれば、がめん を 1つ1つ てで
かきうつす といった めんどうな ことを しなくても すむわけ
です。

 さて、N60-BASIC の がめんを そのまま、プリンタ
にとる めいれい は、つぎ の ように 'LCOPY' という
もの です。 では、 じっさいに やってみましょう。

LCOPY
```



このように，LCOPY という命令は画面のようすをそのまま，プリンタに出すものです．そのままということは，モード1だけでなく，他のモードで表示されたグラフィックによる絵もプリンタに出せるということです．

ただし，プリントアウトは必ずアクティブページですから，モード3と4の画面を LCOPY するには必ずプログラムにしなければなりません．なぜかはよく考えてみてください．



さて，プリンタに関するものには，あと1つあります．それは L POS 関数ですが，見てわかるとおり，カーソルの水平位置を知るため POS 関数のプリンタ用です．プリンタには，カーソルはありませんが，その代わりプリンタヘッドというものがありますので，その位置を知るためのものです．これも試しておいてくださいね．

あっ，忘れていましたが，プリンタにもいろいろな機種がありますが，LCOPY 以外はたいていの機種に同じように使えます．

ただ，LCOPY 命令だけは PC-6000 シリーズのサーマル・プリンタ——PC-6021 のための命令ということを覚えておいてください．

4－3　データの保存

せっかく長い時間かけて，PC にがんばって計算してもらったデータも，PC の電源を切ったとたんに，プログラム同様どこかへ消し飛んでしまいます．かといって，電気を入れっぱなしにはしておけませんし，そのプログラムだけではなく他のプログラムも使いたいですね．

そこで，PC が記憶しているデータを何かの形で保存しておこうとは思いませんか．N60-BASIC には，このような応用をするための外部の機械とデータのやりとりをする命令があります．ここで外部の機械というのは，カセットレコーダ，プリンタ，“RS-232C 通信回線” があります．



まず，カセットレコーダの場合は，なんといっても後で直接 PC に読みこませることができるという点が便利なところです．

もちろん，プリンタの場合は，人間の目に見える形で保存ができますね．

そして，RS-232C 通信回線ですが，これはコンピュータ同志がデータのやりとりをするためのものです．たとえば，きみと友達の PC 同志をつないでやれば，オセロや将棋を2人でやったり，レー

ダー機能で相手をみつける戦争ゲームなどもできるでしょう．なお，これを使うためには，別売のRS-232C ボードが必要なことはいうまでもありません．

これらにデータを送るために，プログラム中で使える命令が PRINT #です。この命令は初級編で出てきた PRINT 命令に“ # ”がついただけですが使い方は少し違います．まず，# のあとに“デバイスナンバー” を指定することによってどの機械にデータを出力するかを決めます．出力するデータは，ナンバーの後にカンマをつけ，その後ろからになります．ただの改行だけを意味する PRINT と同じようにするには，



```
PRINT#n,""
```

のように，ヌルストリングを出力します．さて，このnのところにくるデバイスナンバーは次のとおりです．

0：CRT ディスプレイ（普通の画面）
－1：カセットテープレコーダ
－2：RS-232C 通信回線
－3：プリンタ



#0のときは，単なる PRINT 文と同じですが，この命令の利点は，デバイスナンバーに変数を使えるということです．つまり，変数の値を変えることによって，データの送り先を変えることができます．たとえば，プリンタに出力する前に画面に出してみて確認することが，簡単にプログラムで行えるわけです．プログラムリストからすべての PRINT を LPRINT に変更したり，またその逆をする必要は全く無いのです．

ただし，デバイスナンバーは必ず，0～－3の範囲でなければ，エラーになってしまいます．

さて，データを出しっぱなしでは仕方ありません．データを元通り，PC に入力する命令もあります．そうです，出力が PRINT #なら，入力は“INPUT #”ですね．ただし，プリンタは出力



専用ですから，当然この命令のデバイスナンバーと入力機器の関係は，



0：キーボード
−1：カセットテープレコーダ
−2：RS-232C 通信回線

もちろん，デバイスナンバーに変数を使うことができます．ただし，値が0〜−2の範囲を超えるとエラーとなるのは，いうまでもありません．





第5章　コンピュータのことば

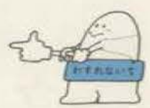
私たちが日頃言葉を使わずには意志の伝達ができないのと同様に，コンピュータも言葉がなくてはどうにも働けません．きみとコンピュータとの言葉は N60-BASIC 言語ですが，コンピュータ内部で使われている言葉はまた別の言葉なのです．

言葉とはいっても，私たちのようなものではありません．もしそうだったら，コンピュータに日本語か英語で話しかけてやれば，プログラムの必要はなくなってしまいますね．なにもわざわざ BASIC 言語を使う必要はないのです．実はコンピュータの言葉というのは数値なのです．コンピュータの中には，"IC"や"LSI"と呼ばれるものがいっぱいあって，それらの間では，数値によってお互いの連絡をとっているのです．

この章では，そのコンピュータの言語を解読するための約束ごとや，直接それを使う方法について説明します．そのため少し難しくなりますから自信のない人は他の専門書等を読んでおくことをおすすめします．

5－1　コンピュータの数の数え方

さて，今まできみの PC はコンピュータの1機種だといってきました．しかし，正確にいうならば，フォン・ノイマン方式のデジタルコンピュータの1機種なのです．



フォン・ノイマンというのは人の名前です．この人が考えた方式だから"フォン・ノイマン方式"というのです．そこで，このフォン・ノイマン方式というのは，どういうものかというと，命令をコンピュータに記憶させる方式です．きみは今まで，BASIC の決まりに基づいてプログラムを作り，それを PC に記憶させて使っていたわけですから，なんだあたりまえのことじゃないかと思うでしょう．

ところが，フォン・ノイマンさん以前のコンピュータはそうではなかったのです．たとえば，極端な例では，1つの仕事しかできない機械だったのです．その機械になにか別の仕事をさせようと思ったら，いちいち機械の配線を組み変えていたのです．

きみのPCは何か別の仕事をさせようと思ったら，プログラムを取り変えてやるだけでいいですね．いかにフォン・ノイマン方式が便利なものかがわかりますね．

そして，デジタルコンピュータです．きみはデジタルというと数字の現れる時計を思いだすでしょう．これに対して針のある時計をアナログ時計なんていいます．デジタルとアナログの時計での違いは，デジタルでは表示されている時間よりも細かい時間はわかりようがないということです．

たとえば，何時何分と表示されているデジタル時計では何秒まではわかりません．しかし，アナログ時計では，秒針がなくても，約何秒というのがわかります．



この"デジタル(Digital)"と"アナログ(Analogue)"というのは，何も時計のことをさす言葉ではありません．コンピュータの世界では，デジタルを計数型，アナログを計量型といっています．

アナログコンピュータでは，内部の電線をどのくらいの電気が流れているかでデータを表しますが，デジタルコンピュータでは電線に電気が流れているか，いないかしか区別しません．



デジタルコンピュータでは，電気が流れている状態を"1"，流れていない状態を"0"として，これを"ビット"といいます．つまり，1ビットは1か0の値を示します．そして，何本かの電線をひとまとまりにしてデータを表すのです．

PCの最も重要な部品に"Z 80"という名のLSI (ICの親分のようなものです) があります．このZ 80は，実はこれだけで立派なコンピュータなのです．そして，これは，"8ビット・マイクロプロセッサ"の1種なのです．難しい言葉が続きましたが，要するにPCの内部では，8ビットを単位として信号が行き来しているのです．



8ビットをまとめて"1バイト"といいます．1ビットで2つの状態が表せますから，1バイトでは2の8乗で256の状態を表せます．これが2×8の16の状態ではないのは，それぞれのビットに"位"がついているからです．つまり，"00000001"と"10000000"では，それぞれ1つのビットだけが1ですが，違うものとして扱われているのです．そして，これを2進数といいます．

普段，私たちの使っているのは10進数です．これは，0から9までは1桁の数で表せますが，9

の次になると1つくり上がって，“1”，“0”の2桁になります．2進数でも同じように，0と1は1桁の数で表せますが，10進数の2になると“10”の2桁になります．つまり，1バイトは8桁の2進数を表すのです．

さて，次のように打ちこんでください．

```
? FRE(0)
```

これは，プログラム用の記憶領域を表示させる命令でしたね．実は，これで表示される値は，PCの“メモリ”（記憶装置）の使われていないものの数です．そして単位はバイトです．つまりPCのメモリは1つが1バイトになっているのです．

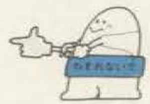
そして，そのビットの状態は2進数で表されるといいました．けれども，そのまま2進数で表したのでは，ビットの状態がよくわかっても，他の数と見比べるととてもわかりにくくなってしまいます．それに，1つのメモリを8桁の数でいちいち表していたのでは面倒ですね．そこで，1バイトを上と下の2つに分けて，4ビットを1つの数で表すと，とても便利なことに気がついた人がいます．すると，4ビットは0～15までを表しますから，16進数で表せばよいわけです．ところが，私たちが日頃使っている10進数では9の次に1桁くり上って2けたになってしまいますから，10～15までをA～Fで表すことにしました．つまり，F(15)の次に1桁くり上って10になるわけです．

たとえば，2進数と10進数と16進数の関係は次のようになります．

$1010_{(2)}=10_{(10)}=A_{(16)}$

$10000_{(2)}=16_{(10)}=10_{(16)}$

$11111111_{(2)}=255_{(10)}=FF_{(16)}$



さて，コンピュータはメモリをいっぱい持っていることはすでにわかっているでしょうが，それらのメモリの1つ1つに名前がついているのです．名前といっても，キミの名前のような立派なものではなく，通し番号なのです．数字の名前なんておかしいですね．そして，これを“アドレス”といい，何番地のメモリという言い方をします．





このアドレスも内部では，16 ビットの 2 進数で示されます．16 ビットですから，2 の 16 乗で 65536 個のアドレスを表すことができます．16 進数で表すと，0 番地から FFFF 番地までとなります．コンピュータではこのように 16 進数をよく使いますから，いちいち 16 進数のいくつという代わりに，“$”を頭につけて表します．つまり，$FF は，10 進数の 255 を意味します．

N 60-BASIC では 16 進数も扱えます．しかし，“$”は文字列のしるしですから，BASIC では，“&H”を付けます．では，次のように打ちこんでみましょう．

```
? &H7FFF
```

リターンキーを押すと，32767 と 10 進数で表示されましたね．これで，“7FFF”を 16 進数として受け付けてくれたことはわかりましたね．



5－2　ビット演算

ここでは，再び 2 部の 3 章で出てきた AND，OR，NOT が出てきます．忘れてしまった人はよく読み返しておいてください．

では，次の例を実行してみてください．

```
? -1 AND 3
 3
```

ちょっとびっくりしましたか？ ANDは論理演算子ですし，左右の式は，ただの数値であって，論理式ではありませんね．それなのにどうしてコンピュータはエラーを出さないのでしょう．出てきた答えも論理値ではありません．故障してしまったのでしょうか．でも，私のPC-6001もキミのPC-6001も全く同じ結果になっているのですから，故障ではありませんよね．

ではそこで，−1を2進数に直してみましょう．2進数の負の数はマイナス記号を使いません．頭のビットを1にするとそれがマイナスの意味を持ちます．そして数値は補数で表します．補数というのは，その数の絶対値のビットを反転して，1をたしたものです．すると−1は

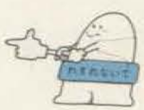
絶対値……00000001
反転………11111110
+1 ………11111111

となるわけです．そうすると，1バイトで表せる整数は127〜−128までということになります．

次に3は2進数では，00000011ですね．そこで，−1 AND 3は，

```
        11111111
AND)    00000011
----------------
        00000011
```

となっているのです．これがビット演算というもので，第3章の論理演算子は実はビットごとに計算していたのです．つまり真の論理値である−1は，全てのビットが1の状態，偽の0は全てのビットが0の状態で，ビット演算の結果は必ず，−1か0になったのです．



そして，ビット演算のANDは，互いに同じ位のビットが両方とも1のときだけ，そのビットが1になるものです．たとえば，



```
         11010110
AND)     01001100
---------------
         01000100
```

となります，これは，10進では，次のようになります．

```
? 214 AND 76
 68
```



同じようにして，ORはどちらかのビットが1のときには，結果もそのビットが1になりますから，たとえば，

```
         11010110
OR )     01001100
---------------
         11011110
```



となります． また，NOTは各ビットを反転させますから，

```
NOT)     11010110
---------------
         00101001
```

となります．

5－3 機械語サブルーチン

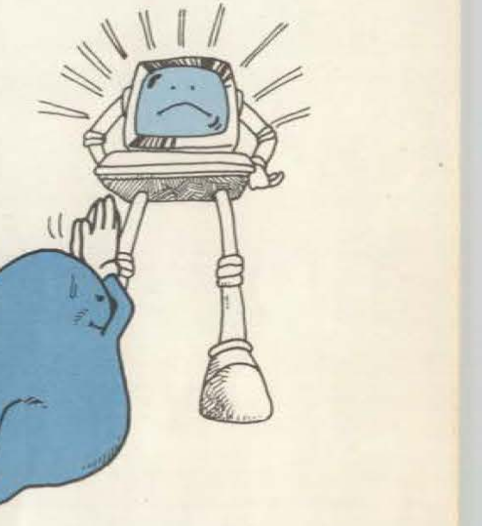
章の頭で説明したコンピュータの言語は，"機械語"と呼ばれています．コンピュータは，BASICという言葉がそのままわかるわけではありませんので，メモリの中に，通訳をやとっています．それが"BASICインタープリタ"というものです．



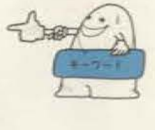
インタープリタは電源を入るとすぐに飛び起きて通訳を始めてくれますので，私たちにとっては

インタープリタの存在がわからなかっただけなのです。

きみは、テレビなどで外人が通訳を通してしゃべっているのを見たことがありますね。外人が1度しゃべったものを通訳が日本語に直してはじめて、外人のいったことがわかるわけです。コンピュータも全く同じです。けれど、通訳を通すとどうしても伝わるまでに時間がかかります。この点でも、コンピュータも全く同じなのです。そこでコンピュータに速くわからせてやる、つまり、高速処理をさせたい時には、きみたちがコンピュータの言葉(機械語)で話してやればよいのです。コンピュータは2進で処理するために、16進で話しかけてやればよいのです。2進と16進の変換はコンピュータがやってくれます。そして、PC-6001の機械語はZ80の機械語を意味します。機械語は人間には非常にわかりにくく、難しいのでそれだけを説明すると1冊の本になってしまいます。ですから、ここではBASICから機械語を呼び出すことだけをお話しします。



では、きみが機械語をマスターしたことにしましょう。それをメモリに入れてやるにはどうやったらいいのでしょう。また、メモリのどこに入れてやればいいのでしょうか。特にその場所には気をつけてやらなくてはいけません。へたなところだと、BASICインタープリタの作業場所を破壊してしまったり、後から入ってきたBASICのプログラムに消されてしまうこともあります。 そこで、まず初めに機械語を入れる場所を確保しなくてはなりません。これはBASICの命令で行うことができます。それはCLEAR命令です。単にこれだけで使うと、変数の消去をします。その次に数値を入れると文字列領域の確保ができますが、小さいプログラムや、文字列をたくさん使わないプログラムでは必要ないでしょう。さて、その次に機械語エリアの先頭番地を指定します。いいかえると、BASICのプログラムの最終アドレスより大きい値をいれてやるわけです。1度指定してしまうと、もう1度CLEARを実行し直さなくてはなりませんから気をつけてください。たとえば次のようにします。



```
CLEAR 300,&HD000
```

では、このようにして、機械語プログラムエリアを確保したところで、さっそく機械語をメモリに入れることにしましょう。アドレスを指定してメモリ上に直接数値を入れる命令はPOKE命令です。入力する数値の範囲は、いうまでもなく1バイトに収まりきる0～255までの整数です。



POKE命令は，BASICから機械語ルーチンに引数を渡す時にも使います．

引数を渡したままでは意味がありませんから，それをメモリ上から読みとることが必要です．この命令はPEEKです．PEEKはアドレスを引数として，そのアドレスのメモリのデータを読みとる関数のようなものです．

では，このようにして，メモリに入れた機械語プログラムをBASICから走らせるためにはどうしたらよいのでしょうか．



BASIC起動状態から機械語サブルーチンを呼び出して実行させる命令が“EXEC”です．EXEC命令はアドレスを指定してやりますが，このアドレスには，変数を使うことも可能です．たとえば，次のような形になります．

```
EXEC &HD000
EXEC  A
```

ところで，機械語サブルーチンを実行する前に注意してほしいことがあります．機械語の場合は，BASICと違って，エラーメッセージを出したりしてくれません．それどころかプログラムのバグによって，PCが全然いうことをきいてくれなくなることもあります．そうなったら，PCの背中の“リセットスイッチ”を押してください．

ただし，このリセットスイッチを押すとちょうど1度電源を切って，入れ直したのと同じですから，プログラムは消えてしまいますから注意してください．



エラーメッセージリスト

?BS Error (Bad Subscript エラー)

★配列の添字が宣言した大きさを超えています.

★配列の添字の個数が宣言したときと違っています.

?CN Error (CoNtinue エラー)

★CONT命令は，プログラムリストを変更した後には実行できません.

?DD Error (Duplicate Definition エラー)

★すでに宣言された配列と同じ名前の配列をDIM命令で宣言しようとしています.

?FC Error (Function Call エラー)

★許されている範囲外の大きさの数を使いました.

COLOR, CONSOLE, CHR$, LEFT$, MID$, RIGHT$, SPC, TAB, ON GOTO, ON GOSUB, LOG, SQR, CLEAR, DIM, KEY

★配列の添字が負になりました.

?ID Error (Illegal Direct エラー)

★ダイレクトモードでは実行できない命令です.

(INPUT, DEF FN の2つ)

?LS Error (Long String エラー)

★文字列の長さが255を超えました.

?MO Error (Missing Operand エラー)

★代入命令の"="の右辺がありません.

★+, *などの演算子があるのに, それで計算するはずの数や文字がありません.

?NF Error (Next without For エラー)

★FORがないのにNEXTが出てきました.

?OD Error (Out of Data エラー)

★READ命令を行おうとしても, もうDATAがありません.

?OM Error (Out of Memory エラー)

★プログラムが長すぎてメモリが足りません.

★配列が大きすぎてメモリが足りません.

★CLEAR命令で文字列用の領域をとりすぎたためメモリが足りません.

★サブルーチンが自分自身をCALLしています.



?OS Error (Out of String space エラー)

★文字列用の領域が足りません. CLEAR命令でもっと増やしてください.

?OV Eror (OVer flow エラー)

★計算の結果が大きすぎます (約1.7E38が限界です).

?RG Error (Return without Gosub エラー)

★GOSUB命令がないのにRETURN命令を実行しています.

?SN Error (SyNtax エラー)

★文字を書き間違えています.

★変数などの名前にアルファベットと数字以外の文字が入っています．
★変数などの名前に，すでにBASICで使われている文字が含まれています．そのような文字はキーワード表にまとめてありますので見てください．

?ST Error (String Too complex エラー)
★文字列の計算が複雑すぎます．

?TM Error (Type Mismatch エラー)
★文字列と数をいっしょに取り扱ってはいけません．

?TR Error (Tape Read エラー)
★テープの読みこみが正しく行われませんでした．

?UF Error (Undefined Function エラー)
★D E F F N 命令で定義していないのにFNA(X)などの“FN”で始まる関数を使っています．

?UL Error (Undefined Line number エラー)
★GOTO命令，GOSUB命令，RESTORE命令，RUN命令で指定した行番号のついている行がありません．
★プログラムを訂正しているときに，ない行を消そうとしました．

?/0 Error (division by zero エラー)
★0でわり算をしようとしました．
★TAN(X)を計算するときに，引数Xがちょうど$\pi/2$(1.570796326)になりました．

キーワード表

ABS	KEY	REM
AND	LCOPY	RESTORE
ASC	LEFT$	RETURN
CHR$	LEN	RIGHT$
CLEAR	LET	RND
CLOAD	LINE	RUN
CLS	LIST	SCREEN
COLOR	LLIST	SGN
CONSOLE	LOCATE	SIN
CONT	LOG	SOUND
COS	LPOS	SPC
CSAVE	LPRINT	SQR
CSRLIN	MID$	STEP
DATA	NEW	STICK
DEF	NEXT	STOP
DIM	NOT	STRIG
END	ON	STR$
EXEC	OR	TAB
EXP	OUT	TAN
FN	PAINT	THEN
FOR	PEEK	TIME
FRE	PLAY	TO
GOSUB	POINT	USR
GOTO	POKE	VAL
IF	POS	+
INKEY$	PRESET	*
INP	PRINT	/
INPUT	PSET	−
INT	READ	^

さくいん

タ

ナ

ハ

NEC

日本電気株式会社
パーソナルコンピュータ事業部パーソナルコンピュータ販売部
東京都港区芝5丁目33番7号(徳栄ビル) 電話 東京(03)453-5511(大代)〒108

新日本電気株式会社
パーソナルコンピュータ・ディスプレイ事業部販売部
川崎市高津区久本210番地 電話 (044)833-5201(大代)〒213

N60-BASIC

価格2,500円 08227048